République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Dr. Tahar Moulay de Saida
Faculté de Technologie

Département d'Electrotechnique

Polycopié de cours

Intitulé :

Automate Programmable Industriel

Présenté par :

Dr. LABANE Chrif

Ce cours est destiné aux étudiants de 3éme année Licence Automatique

Septembre 2024

Avant propos

Ce polycopié de cours est destiné aux étudiants de troisième année Licence en Automatique. Ce cours (Automate Programmable Industrielle) vise à mettre entre les mains de nos étudiants un document didactique leurs permettant d'acquérir les notions de base nécessaires à une bonne compréhension des systèmes automatisés, les Automates Programmables Industriels et la modélisation d'un système automatisé en utilisant le GRAFCET. Le cours est organisé en quatre chapitres, le premier concerne généralités sur les systèmes automatisé détaillée et les différentes parties des systèmes automatisés. Puis la modélisation du système automatisé sous le GRAFCET, cet outil simple et efficace est le plus utilisé dans le monde industriel. Ensuite l'étude des automates programmables industriels et ses différentes parties détaillées serons présentées. Nous achevons ce cours par la programmation des automates industrielle.

Saida le 27/09/2024

Sommaire

Chapitre 1 : Généralités sur les systèmes automatisés

Chapitre 2 : Le grafcet

Chapitre 3 : Architecture des API

Chapitre 4 : Programmation des API

1.1 Définition

Un automatisme est un sous-ensemble des machines, destinée à remplacer l'action de l'être humain dans des taches en générale simples et répétitives, réclamant précision et rigueur. On passe d'un système dit manuel, à un système mécanisé, puis au système automatisé. Dans l'industrie, les automatismes sont devenus indispensables : ils permettent d'effectuer quotidiennement les taches les plus ingrates, répétitives et, dangereuses. Parfois, ces automatismes sont d'une telle rapidité et d'une telle précision, qu'ils réalisent des actions impossibles pour un être humain. L'automatisme est donc synonyme de productivité et de sécurité.

1.2 Notions d'automatisme

-**Système :** Ensemble de composants issus de différentes technologies (mécanique, électronique, informatique.etc.) qui assemblés d'une certaine façon à former une fonction.
- **Automatique :** C'est l'ensemble des sciences et des techniques utilisées dans la conception et la réalisation des Systèmes Automatisés (SA).
- **Procédé :** une machine, un ensemble de machines ou plus généralement un équipement industriel.
- **Chaines opérationnelles :** ensemble de procédés fonctionnent séquentiellement pour faire une tache spécifiques.
- **Automatisation d'un procédé :** consiste à en assurer la conduite par un dispositif technologique (appelé automatisme). L'ensemble procédé est appelé système automatisé.
- **Opérateur :** est la personne qui va faire fonctionner le système.
- **Pré-actionneurs :** Composant contrôlable utilisé pour distribuer l'énergie aux actionneurs.
- **Informatique Industrielle :** Une branche de l'informatique appliquée qui couvre l'ensemble des techniques de conception et de programmation de systèmes informatisés à vocation industrielle qui ne sont pas des ordinateurs.
- **Cahier de charge :** est le descriptif fourni par l'utilisateur au concepteur de l'automatisme pour lui indiquer les différents modes de marches et les sécurités que devra posséder l'automatisme. Ce cahier des charges représente le comportement de la partie opérative par rapport `a la partie commande.

1.3 Exemple d'une automatisation industrielle

La palettisation fait partie des systèmes de manutention qui se sont le plus développés au cours des trois dernières décennies. Elle consiste à grouper un certain nombre de colis sur un support : la palette ; l'opération de groupage est faite par un palettiseur comme montre la figure suivante.

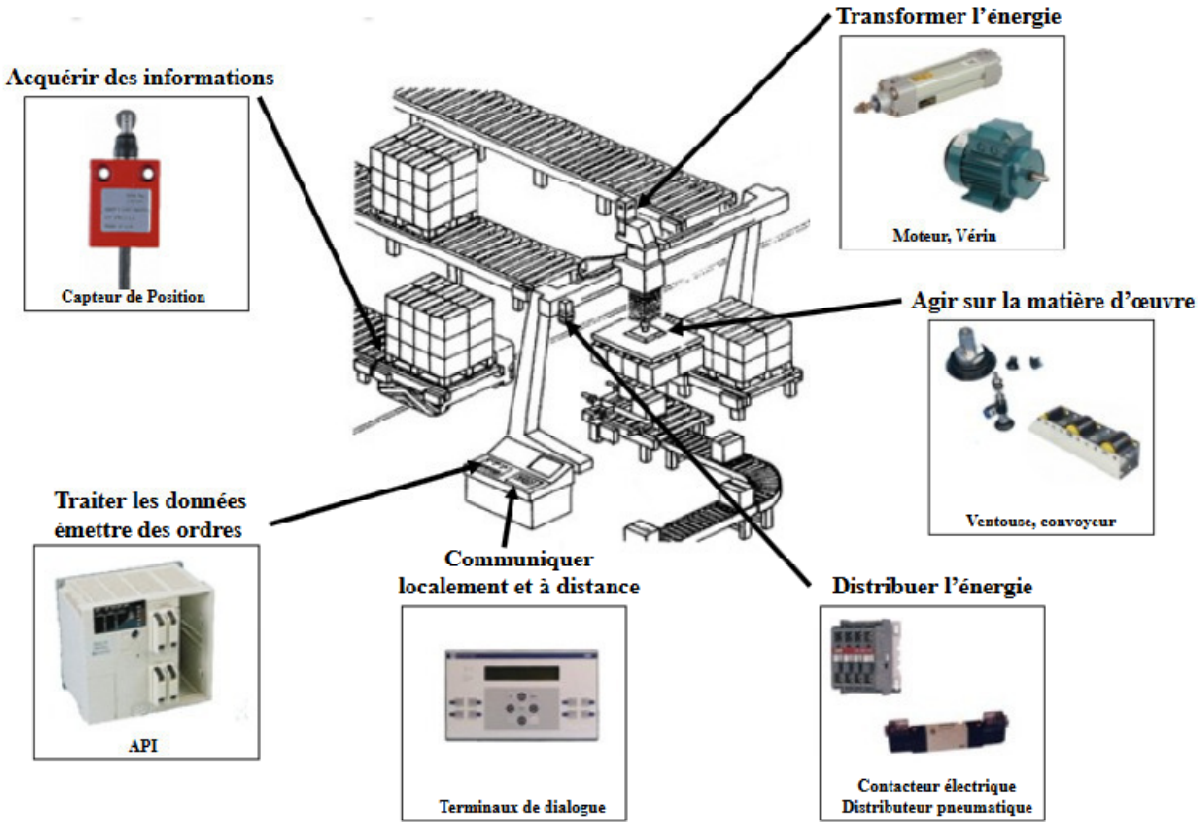


Fig. 1.1 Exemple d'une Chaine de palettisation.

1.4 Objectif de l'automatisation

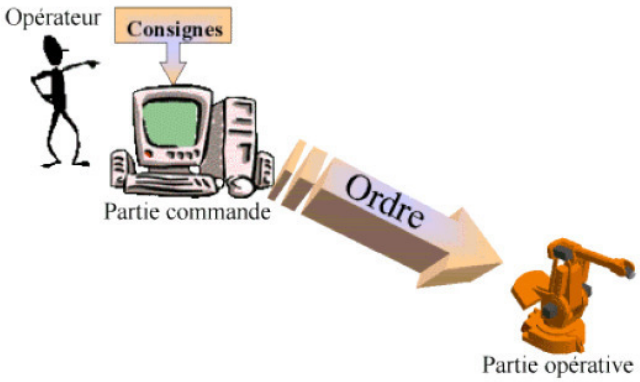
Le but principale est de remplacer l'intervention humaine dans la plus part des taches qui peut êtres rapides, très précises, dangereuses, répétitives et pénibles.
– L'automatisation aide à la compétitivité du produit.
– Améliorer la productivité en réduisant les couts de production.
– Améliorer la qualité de production.
– Augmenter la production.
– Améliorer la flexibilité du système de production.
– Améliorer la qualité et la disponibilité des produits.

1.5 Modes de fonctionnement des automatismes

Les degrés d'automatisation d'un système change selon la nature, la complexité et les objectifs assignés au projet. Il existe 3 modes de fonctionnement d'un automatisme :

1.5.1 Mode Surveillance

Dans ce mode, l'automatisme a une fonction passive vis-à-vis du procédé qu'il pilote. L'organe de contrôle acquit les informations et les analyser pour fournir des journaux de bord. L'objectif dans ce mode est la connaissance technique et économique du procédé.

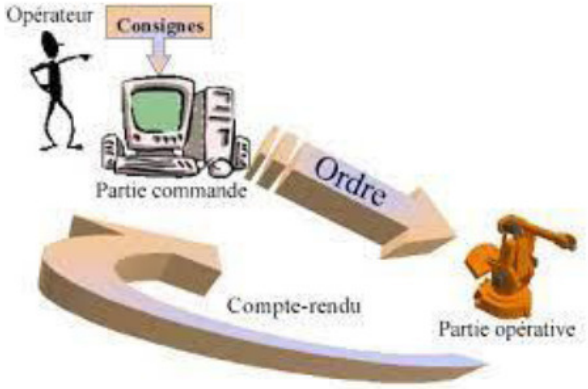


1.5.2 Mode Guide-opérateur

Les traitements sont plus élaborés que dans le cas précédent. L'automatisme propose des actions pour conduire le procédé selon un critère donné. L'automatisme ne réagit pas directement sur le procédé, il a donc un fonctionnement en boucle ouverte.

1.5.3 Mode Commande

L'automatisme à une structure en boucle fermée. On a une automatisation complète de certaines fonctions. L'acquisition des informations, leurs traitement et enfin l'action sur le procédé.



Mode de fonctionnement	Acquisition	Traitement	Action	Structure
Surveillance	×			Boucle ouverte
Guide-opérateur	×	×		Boucle ouverte
Commande	×	×	×	Boucle fermée

Tab. 1 : Différents modes de fonctionnement d'un automatisme

1.6 Structure d'un système automatisé

Ce chapitre permet de comprendre la structure d'un Système Automatisé de Production et de définir les différentes parties de ce système. Un système de production est dit automatisé

lorsqu'il peut gérer de manière autonome un cycle de travail préétabli qui se décompose en séquences et/ou en étapes. Les systèmes automatisés, utilisés dans le secteur industriel, possèdent une structure de base identique. Ils sont constitués de plusieurs parties plus ou moins complexes reliées entre elles :
– la partie opérative (PO) ;
– la partie commande (PC) ou système de contrôle/commande (SCC) ;
– la partie relation (PR) de plus en plus intégrée dans la partie commande.

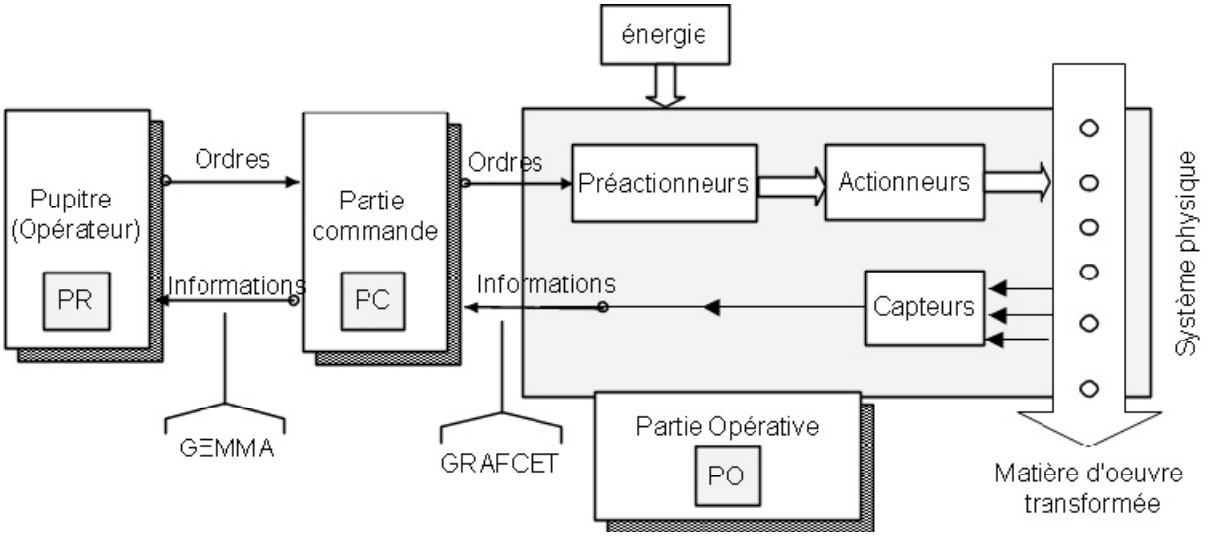


Fig. 1.2 Structure d'un système automatisé de production

1.7 Description des différentes parties

1.7.1 La partie opérative

C'est la partie visible du système. Elle comporte les éléments du procédé, c'est à dire :
– des pré-actionneurs (distributeurs, contacteurs) qui reçoivent des ordres de la partie commande ;
– des actionneurs (vérins, moteurs, vannes) qui ont pour rôle d'exécuter ces ordres. Ils transforment l'énergie pneumatique (air comprimé), hydraulique (huile sous pression) ou électrique en énergie mécanique ;
– des capteurs qui informent la partie commande de l´exécution du travail.
Par exemple, on va trouver des capteurs mécaniques, pneumatiques, électriques ou magnétiques montés sur les vérins. Le rôle des capteurs (ou détecteurs) est donc de contrôler, mesurer, surveiller et informer la PC sur l'´evolution du système.

1.7.2 La partie commande

Ce secteur de l'automatisme gère selon une suite logique le déroulement ordonné des opérations à réaliser. Il reçoit des informations en provenance des capteurs de la Partie Opérative, et les restitue vers cette même Partie Opérative en direction des pré-actionneurs et actionneurs. L'outil de description de la partie commande s'appelle le GRAphe Fonctionnel de Commande ´Etape / Transition (GRAFCET).

1.7.3 La partie relation

Sa complexité et sa taille dépendent de l'importance du système. EIle regroupe les différentes commandes nécessaires au bon fonctionnement du procédé : marche-arrêt, arrêt d'urgence, marche automatique, marche cycle/cycle… etc.

1.8 Différents types de commande

Les systèmes automatisés, utilisés dans le secteur industriel deux types de système de commande : le système automatisé combinatoire et le système automatisé séquentiel.

1.8.1 Le système automatisé combinatoire

Ces systèmes n'utilisent aucun mécanisme de mémorisation : à une combinaison des entrées ne correspond qu'une seule combinaison des sorties. La logique associée est la logique combinatoire. Les outils utilisés pour les concevoir sont l'algèbre de Boole, les tables de vérité, les tableaux de Karnaugh.
Les systèmes automatisés utilisant la technique "combinatoire" sont aujourd'hui très peu utilisés. Ils peuvent encore se concevoir sur des mécanismes simples où le nombre d'actions à effectuer est limité. Ils présentent en outre l'avantage de n'utiliser que très peu de composants

1.8.2 Le système automatisé séquentiel

Ces systèmes sont les plus répandus dans le domaine industriel. Le déroulement du cycle s'effectue étape par étape. A une situation des entrées peuvent correspondre plusieurs situations de sortie. La sélection d'une étape ou d'une autre dépend de la situation antérieure du dispositif.
La logique associée est appelée logique séquentielle. Elle peut être avec commande :

– technologie électronique;
– technologie électrique;
– technologie pneumatique;

1.8.2.1 La logique programmée : commande électrique

L'élément principal s'appelle l'Automate Programmable Industriel ou l'API. La détection est électrique. Le pilotage des actionneurs se fait par l'intermédiaire de relais ou de distributeurs. Il existe sur le marché de nombreuses marques d'automates : Télémécanique, Siemens, Omron, Allen Bradley, Cegetel, etc. . .

1.8.2.2 La logique câblée : commande pneumatique

L'élément principal s'appelle module séquenceur et l'association de modules constitue un ensemble appelé séquenceur. La détection est pneumatique, le pilotage des distributeurs se fait par une action de l'air comprimé sur un piston qui fait déplacer le tiroir du distributeur à droite ou à gauche. L'ensemble, appelé tout pneumatique, est homogène et fiable.

1.8.3 Les systèmes asservis

Pour ces systèmes, on désire que la sortie suive avec précision les variations de l'entrée, et ceci avec un temps de réponse réduit. C'est par exemple le cas avec une direction assistée d'automobile ou la commande des gouvernes d'un avion. Applications : les robots industriels.

1.9 Domaines d'application des systèmes automatisés

Aujourd'hui, il serait difficile de concevoir un système de production sans avoir recours aux différentes technologies et composants qui forment les systèmes automatisés.

1.9.1 Les avantages

La capacité de production accélérée ;
L'aptitude à convenir à tous les milieux de production ;
La souplesse d'utilisation ;
La création de postes d'automaticiens.

1.9.2 Les inconvénients

Le coût élevé du matériel, principalement avec les systèmes hydrauliques.
La maintenance doit être structurée.
La suppression d'emplois.

1.10 Exemples de domaines d'application

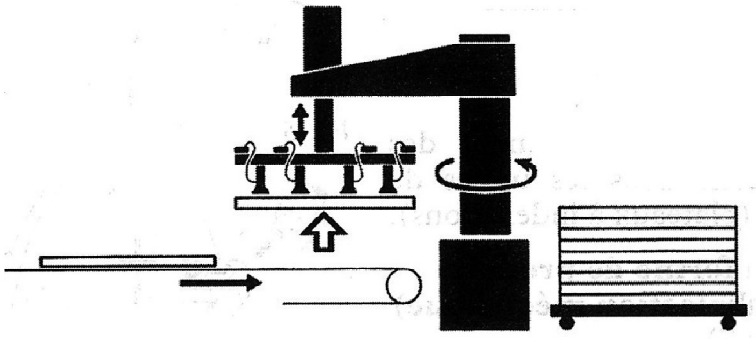


Fig. 1.3 – Le conditionnement, par ex. le déplacement d'objets suivant un angle quelconque ou le conditionnement sur palette après emballage

Fig. 1.4 L'industrie automobile avec l'utilisation de robots industriels pour effectuer l'assemblage et la peinture des carrosseries

Fig. 1.5 L'industrie du bois avec les opérations de débit, de sciage et d'usinage du bois

Fig. 1.6 Machine outil : l'automatisation est ici assez importante. L'une des principales applications est dans les unités de perçage.

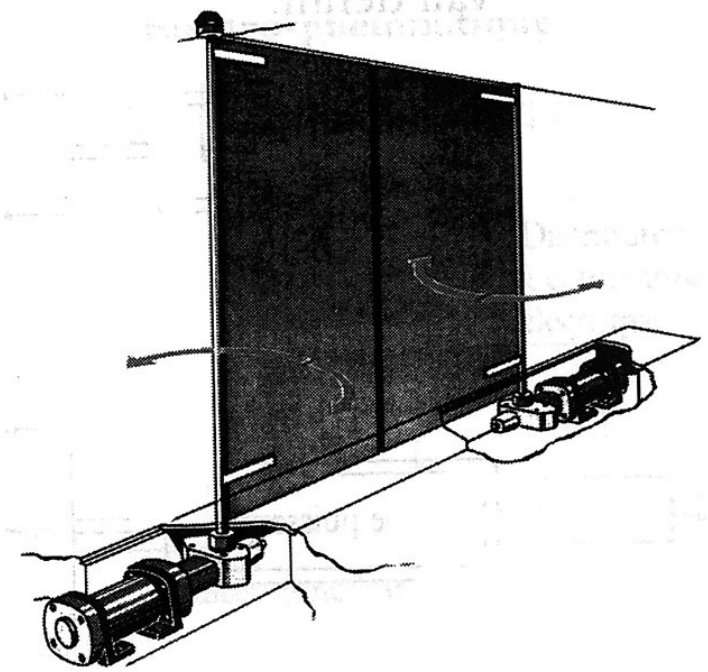


Fig. 1.7 Automatisation de services : ouvertures programmées de portes et fenêtres, gestion centralisée de bâtiment.

Fig. 1.8 Contrôle de produits : Détection de défauts en bout de chaine de production

1.11Les périphériques

On distingue trois catégories de périphériques :

1.11.1 Actionneurs

Les Actionneurs permettent de transformer l'énergie reçue en un phénomène physique (déplacement, dégagement de chaleur, émission de lumière ...etc.). Il existe différents types actionneurs utilisé dans un système automatisés, la majorité des actionneurs sont classés selon la nature de leur énergie d'alimentation.

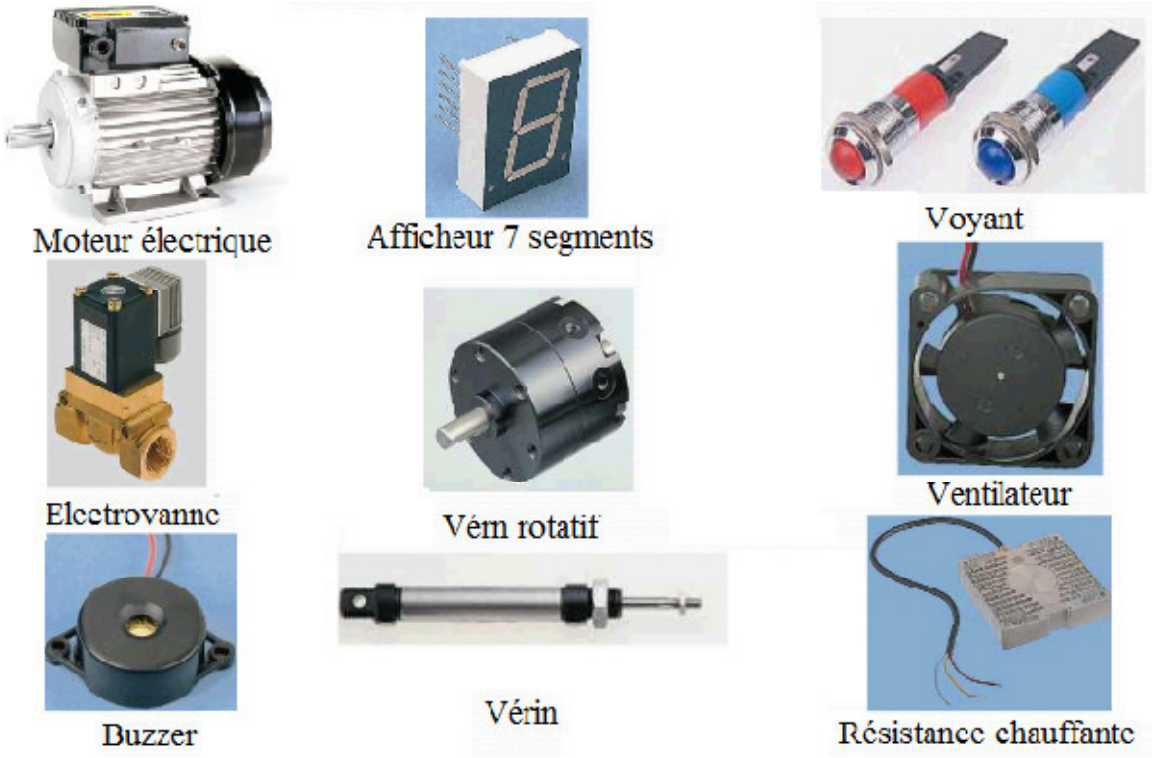


Fig. 1.9 Exemples d'actionneurs.

1.11.2 Capteurs

Les Capteurs permettent de prélever sur la partie opérative, l'état de la matière d'œuvre et son évolution ; il est capable de détecter un phénomène physique dans son environnement (déplacement, présence, chaleur, lumière, pression...) puis convertir l'information physique en une information codée compréhensible par la partie commande. Peuvent être électriques ou pneumatiques. Signaux du type TOR, Analogique ou Numérique.

Fig. 1.10 Différents types de capteurs.

1.11.3 Effecteur

Il est un organe utilise l'énergie convertie par les actionneurs pour produire un effet utile. Dans une chaine d'action, l'effecteur est l'organe terminal qui agit directement sur le produit traité par le système. Ci-dessous un exemple d'un effecteur.

Fig. 1.11 Organe effecteur d'un robot industriel.

1.11.4 Unité de dialogue

Elle permet à l'opérateur d'envoyer des consignes à l'unité de traitement et de recevoir de celle-ci des informations sur le déroulement du processus. En utilisant les pupitres de commande industriels qui offrent des services performants d'affichage, de saisie de paramétrage, de commande pour la conduite de la machine, mais aussi de gestion de défauts, d'historiques sur les pièces et les défauts.

Fig. 1.12 Pupitre de commande industrielle.

2.1 Introduction

Le GRAFCET **(Graphe Fonctionnel de Commande Etape/Transition)** a été proposé par ADEPA **(agence pour le développement de la Productique Appliquée à l'industrie**) et l'AFCET **(Association Française pour la Cybernétique Economique et Technique)** en 1977, et normalisé en 1982 par la NF C03-190. C'est un outil graphique destiné à décrire les différents comportements de l'évolution d'un automatisme séquentiel. Il est très utilisé pour la programmation des automates programmables industriels. Lorsque le mot grafcet est écrit en minuscule, il fait alors référence à un modèle obtenu à l'aide des règles du GRAFCET.

Le GRAFCET possède les avantages suivants :

- Il est indépendant de la matérialisation technologique.
- Il traduit de façon cohérente le cahier des charges.
- Il est bien adapté à la complexité des systèmes automatisés et à la conception et réalisation.

Dans ce chapitre, on commence par donner les bases nécessaires de la modélisation des systèmes automatisés en langage GRAFCET. Ensuite, on illustre ces différentes règles et on présente les différentes structures du grafcet. Les différentes actions associées aux étapes sont définies par la suite. Enfin, on montre la transformation du modèle grafcet en des équations.

2.1.1 Exemple introductif On veut contrôler une chaine de fonctionnement contient deux vérins pneumatiques (V1 et V2).

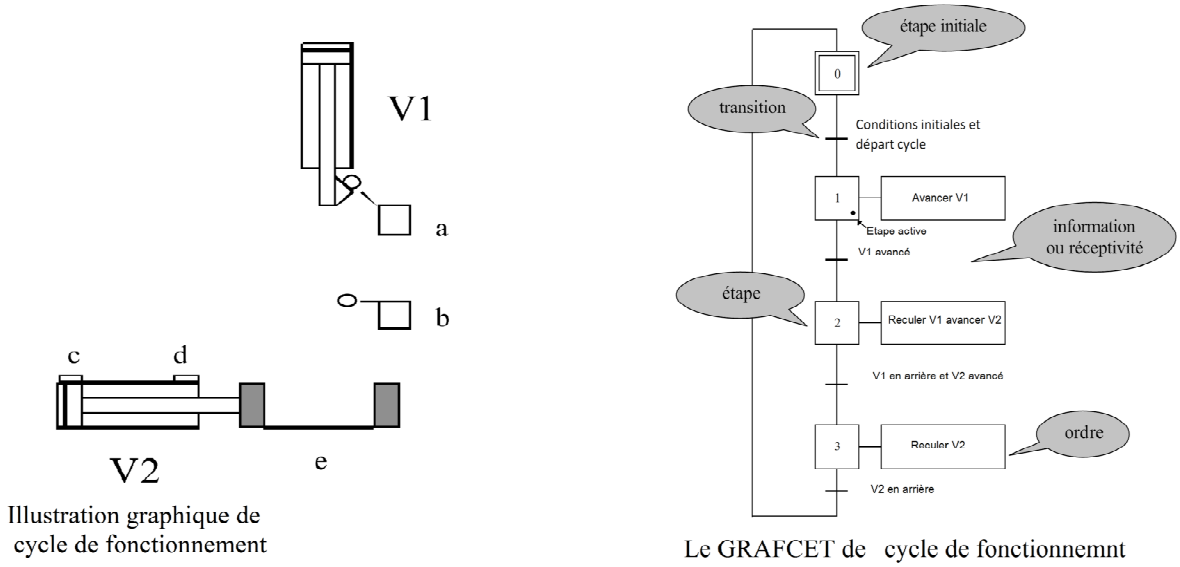


Illustration graphique de cycle de fonctionnement

Le GRAFCET de cycle de fonctionnemnt

Fig. 2.1 Exemple de GRAFCET simple.

2.2 Description du GRAFCET

2.2.1 Les étapes

L'étape symbolise un état ou une partie de l'état du système. L'étape possède deux états possibles : active représentée par un jeton dans l'étape ou inactive. L'étape i, repérée numériquement, possède ainsi une variable d'´etat, appelée variable d'étape Xi. Cette variable est une variable booléenne valant 1 si l'étape est active, 0 sinon.
La situation initiale d'un système automatisé est indiquée par une étape dite étape initiale et représentée par un carré double.

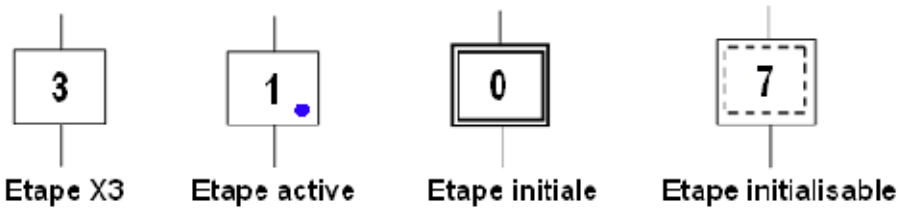


Fig. 2.2. Situation des étapes d'un système automatisé

2.2.2 Actions associées aux étapes

chaque étape est associée une action ou plusieurs, c'est à dire un ordre vers la partie opérative ou vers d'autres grafcets. Mais on peut rencontrer aussi une même action associée à plusieurs étapes ou une étape vide (*sans action*).

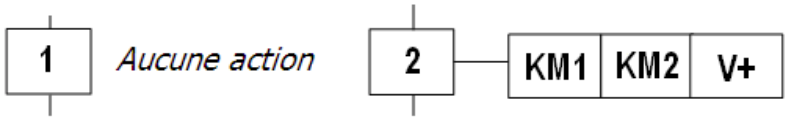


Fig. 2.3. Actions associées aux étapes

2.2.3 Transition

Une transition indique la possibilité d'évolution qui existe entre deux étapes et donc la succession de deux activités dans la partie opérative. Lors de son franchissement, elle va permettre l'évolution du système. A chaque transition est associée une condition logique appelée réceptivité qui exprime la condition nécessaire pour passer d'une étape à une autre.

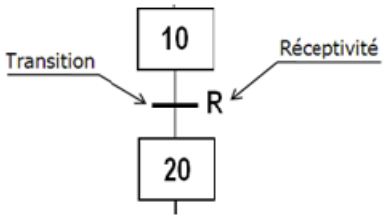


Fig. 2.4. Transition associée une condition logique appelée réceptivité

La réceptivité est une information d'entrée qui est fournie par :
l'opérateur : pupitre de commande,
la partie opérative : états des capteurs, du temps, d'un comptage ou toute opération logique, arithmétique...
du grafcets : d'autres grafcet pour la liaison entre grafcets ou de l'état courant des étapes du grafcet (les Xi),
d'autres systèmes : dialoguent entre systèmes, ...

Remarque: Si la réceptivité n'est pas précisée, alors cela signifie qu'elle est toujours vraie (=1).

2.2.4 Liaisons orientées

Elles sont de simples traits verticaux qui relient les étapes aux transitions et les transitions aux étapes. Elles sont normalement orientées de haut vers le bas. Une flèche est nécessaire dans le cas contraire.

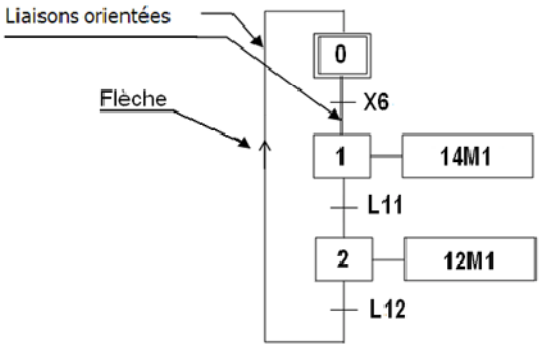


Fig. 2.5. Liaisons orientées entre les étapes et les transitions

2.3 Classification des actions associées aux étapes

L'action associée à l'étape peut être de 3 types : **continue**, **conditionnelle** ou **mémorisée**. Les actions peuvent être classées en fonction de leur durée par rapport à celle de l'étape.

2.3.1 Actions continues :

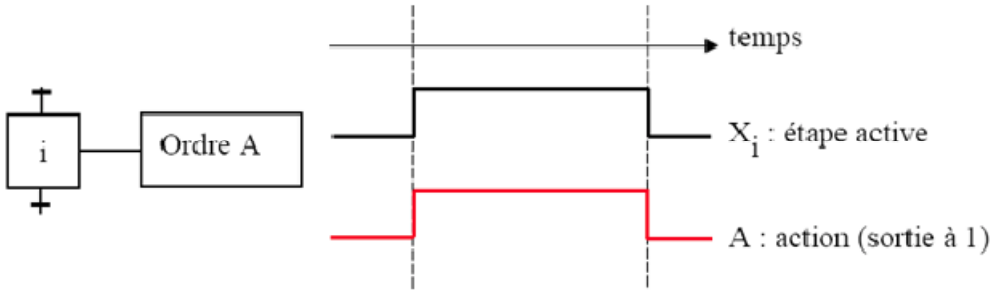


Fig. 2.6. Chronogramme de l'action continue

2.3.2 Actions conditionnelles:

Une action conditionnelle n'est exécutée que si l'étape associée est active et si la condition associée est vraie. Elles peuvent être décomposées en 3 cas particuliers:

2.3.3 Action conditionnelle simple : Type C

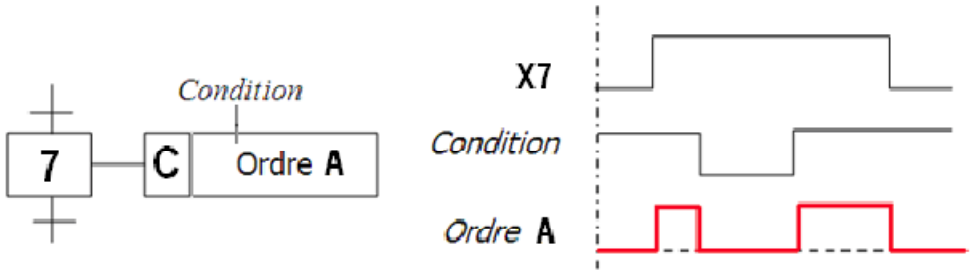


Fig. 2.6. Chronogramme de l'action conditionnelle simple

2.3.4 Action retardée : Type D (delay)

Le temps intervient dans cet ordre conditionnel comme condition logique. L'indication du temps s'effectue par la notation générale " t / xi / q " dans laquelle "xi" indique l'étape prise comme origine du temps et "q" est la durée du retard.

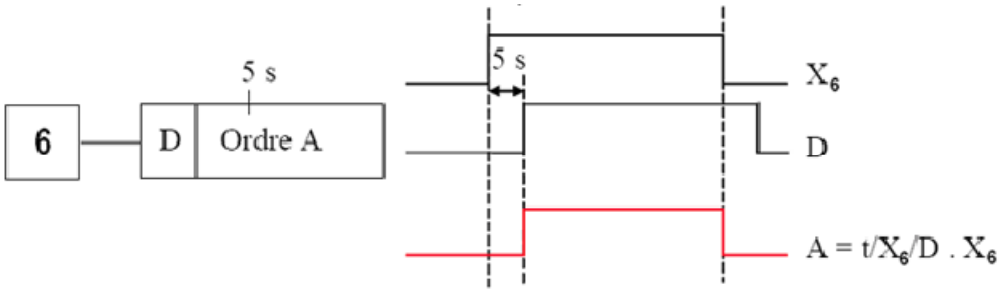


Fig. 2.7. Chronogramme de l'action retardée

Exemple :

2.3.5 Action de durée limitée: Type L (limited)

"t /x6/ 5s" : prendra la valeur logique 1, 5s après la dernière activation de l'étape 6.
L'ordre est émis dès l'activation de l'étape à laquelle il est associé ; mais la durée de cet ordre sera limitée à une valeur spécifiée.

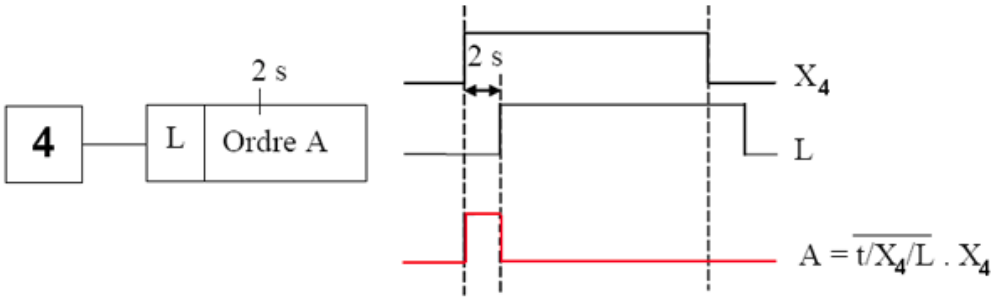


Fig. 2.8. Chronogramme d'action de durée limitée

L'ordre "A" est limité à 2s après l'activation de l'étape 4.

2.3.6 Action maintenue sur plusieurs étapes:

Afin de maintenir la continuité d'une action sur plusieurs étapes, il est possible de répéter l'ordre continu relatif à cette action, dans toutes les étapes concernées ou d'utiliser une description sous forme de séquences simultanées (*Les séquences simultanées seront traitées ultérieurement*).

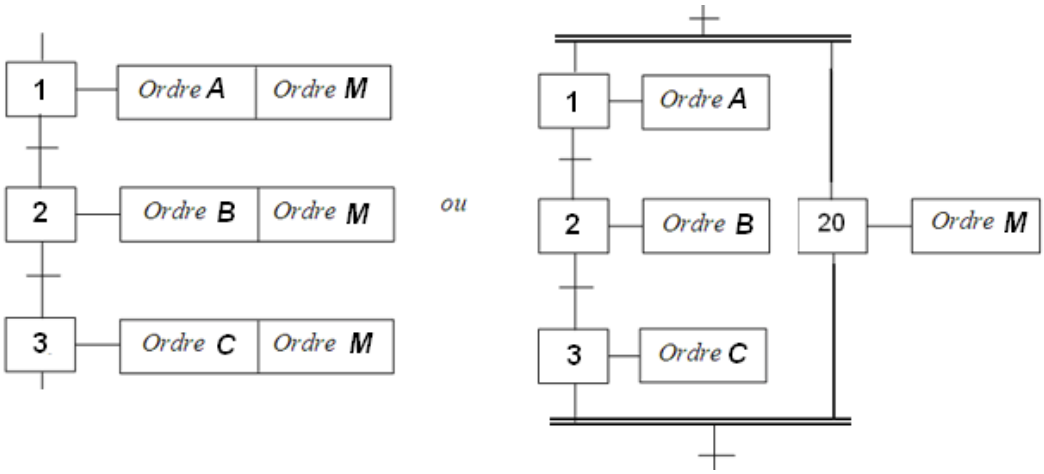


Fig. 2.9. Action maintenue sur plusieurs étapes

2.3.7 Action mémorisée :

Le maintien d'un ordre, sur la durée d'activation de plusieurs étapes consécutives, peut également être obtenu par la mémorisation de l'action, obtenue par l'utilisation d'une fonction auxiliaire appelée fonction mémoire.

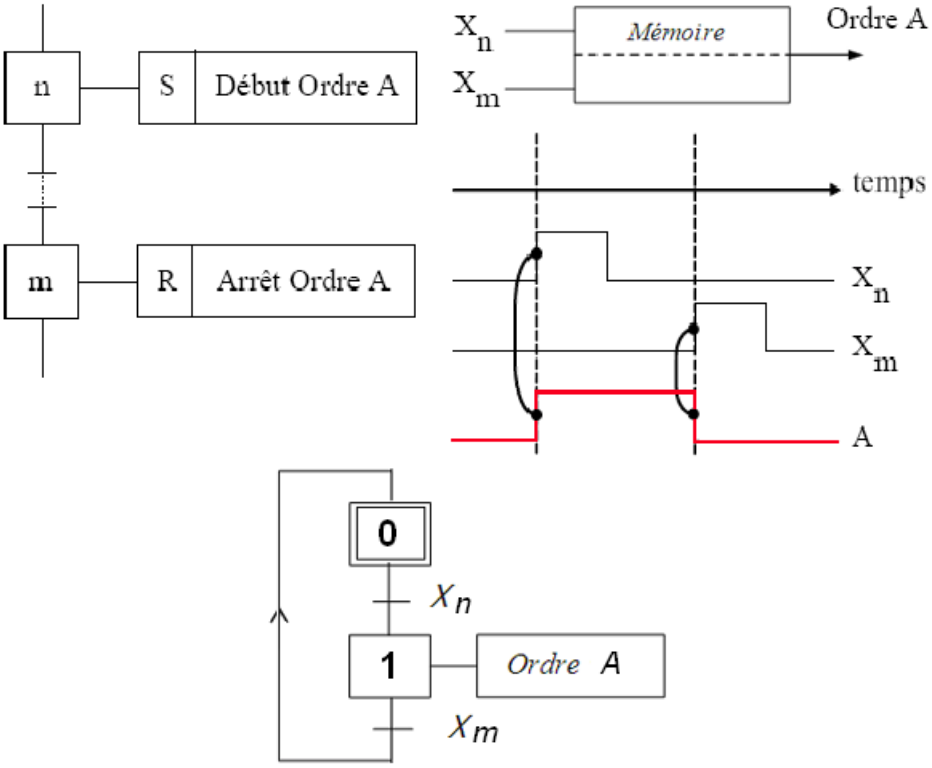


Fig. 2.10. Action à effet Maintenue par une Action Mémorisée

2.4 Règles d'évolution d'un GRAFCET

Règle N°1 : ***Condition initiale.***
A l'instant initial, seules les étapes initiales sont actives.
Règle N°2 : ***Franchissement d'une transition***.
Pour qu'une transition soit validée, il faut que toutes ses étapes amont (immédiatement précédentes reliées à cette transition) soient actives. Le franchissement d'une transition se produit lorsque la transition est validée, et seulement si la réceptivité associée est vraie.

Règle N°3 : ***Evolution des étapes actives***.

Le franchissement d'une transition entraîne obligatoirement l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.

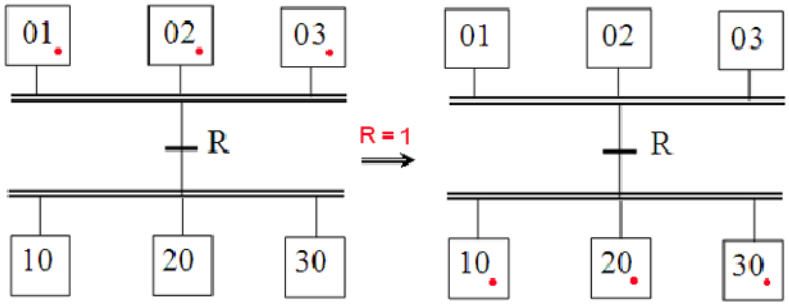


Fig. 2.11. Franchissement d'une transition

Règle N°4 : ***Franchissement simultané***.
Toutes les transitions simultanément franchissables à un instant donné sont simultanément franchies.

Règle N°5 : ***Conflit d'activation.***
Si une étape doit être simultanément désactivée par le franchissement d'une transition aval, et activée par le franchissement d'une transition amont, alors elle reste active. On évite ainsi des commandes transitoires (néfastes à la partie opérative).

2.4.1 Sélection de séquence et séquences simultanées :

Le méta-modèle grafcet présente deux structures particulières, la sélection de séquence et les séquences simultanées.

2.4.2 Sélection de séquence

La sélection de séquence dans un grafcet permet de choisir une suite d'étapes plutôt qu'une autre. Cette structure est composée d'une seule étape en amont et de plusieurs transitions en aval qui permettront le choix de la séquence. Elle se représente à l'aide d'un simple trait horizontal. La fin d'une sélection de séquence permet la reprise d'une séquence unique.

Exemples

- début de sélection de séquence :

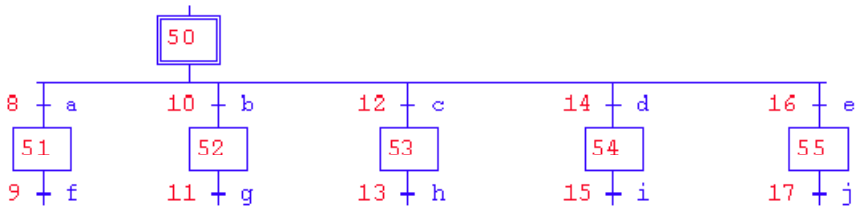


Fig. 2.13. Début de sélection de séquence

- fin de sélection de séquence :

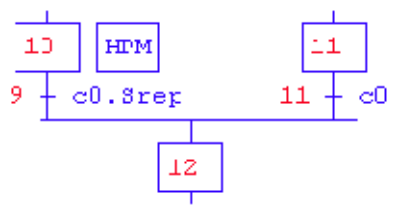


Fig. 2.14. Fin de sélection de séquence

2.4.3 Séquences simultanées

Lorsque l'on souhaite réaliser plusieurs séquences simultanément on utilise cette structure. Elle est composée d'une seule étape et d'une seule transition en amont qui permet de déclencher simultanément plusieurs séquences d'étapes. Elle se représente à l'aide d'un double trait horizontal. A la fin d'une série de séquences simultanées on retrouve, en général, un double trait suivi d'une seule transition. Attention, c'est le déclenchement des étapes situées immédiatement après le double trait qui est simultané et non l'évolution des séquences. L'évolution de chacune des séquences d'étapes dépendra des conditions d'évolution et des actionneurs utilisés sur la Partie Opérative (compteur, temporisations, états de capteurs,...).

Exemples :

Début de séquences simultanées :

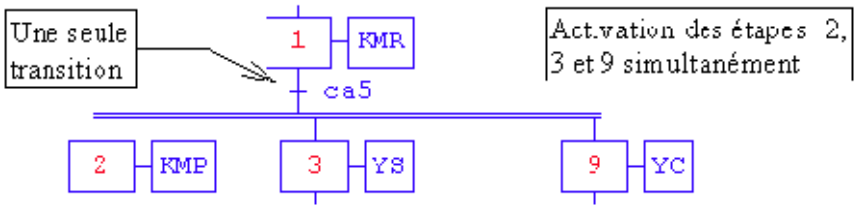


Fig. 2.15. Début de séquences simultanées

Fin de séquences simultanées :

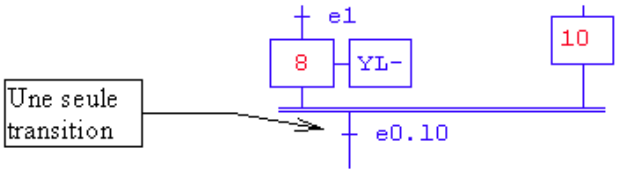


Fig. 2.16. Fin de séquences simultanées

2.4.5 Saut d'étapes et reprise de séquence

Le saut d'étapes permet de sauter une ou plusieurs étapes lorsque les actions associées sont inutiles à réaliser, La reprise de séquence (ou boucle) permet de reprendre, une ou plusieurs fois, une séquence tant qu'une condition n'est pas obtenue.

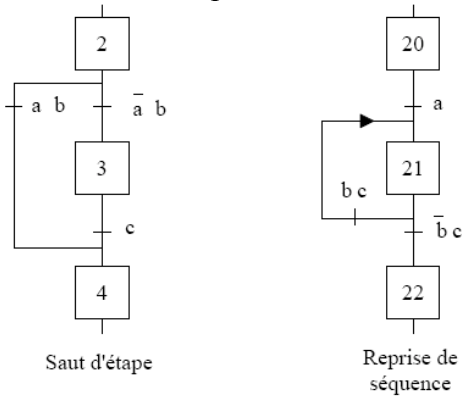


Fig. 2.17. Saut d'étapes et reprise de séquence

2.4.6 Aiguillage entre deux ou plusieurs séquences (Divergence en OU)

On dit qu'il y a un aiguillage ou divergence en OU lorsque le grafcet se décompose en deux ou plusieurs séquences selon un choix conditionnel. Comme la divergence en OU on rencontre aussi la convergence en OU. On dit qu'il y a convergence en OU, lorsque deux ou plusieurs séquences du grafcet converge vers une seule séquence.

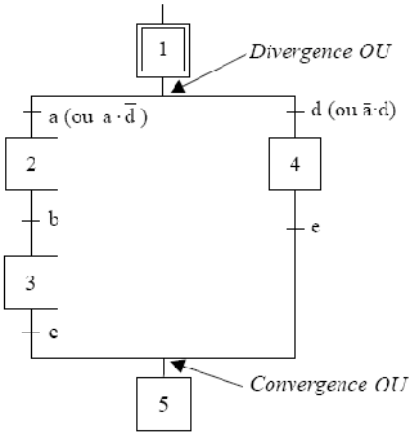


Fig. 2.18. Aiguillage entre deux ou plusieurs séquences

Si les deux conditions « a » et « d » sont à 1 simultanément, les étapes 2 et 4 vont devenir actives simultanément, situation non voulue par le concepteur. Donc elles doivent être des conditions exclusives

2.4.7 Parallélisme entre deux ou plusieurs séquences (ou séquences simultanées ou divergence convergence en ET) :

Au contraire de l'aiguillage, où ne peut se dérouler qu'une seule activité à la fois, On dit qu'on se trouve en présence d'un parallélisme structurel, si plusieurs activités indépendantes pouvant se dérouler en parallèle. Le début d'une divergence en « ET » et la fin d'une convergence en « ET » d'un parallélisme structurel sont représentés par deux traits parallèles

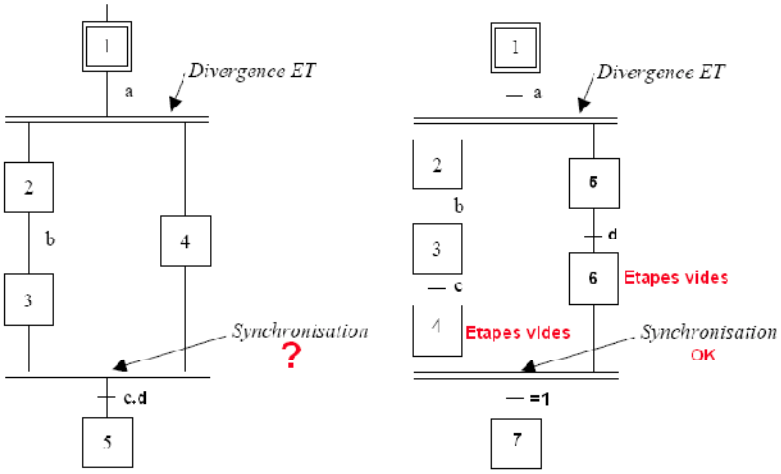


Fig. 2.19. Parallélisme entre deux ou plusieurs séquences

La synchronisation permet d'attendre la fin de plusieurs activités se déroulant en parallèle, pour continuer par une seule.

2.5 Liaison entre grafcets :

Une étape dans un grafcet peut servir comme réceptivité à une autre étape d'un autre grafcet. Cette méthode est utilisée aussi pour synchroniser deux grafcets c'est à dire rendre l'évolution de l'un dépendent de l'évolution de l'autre.

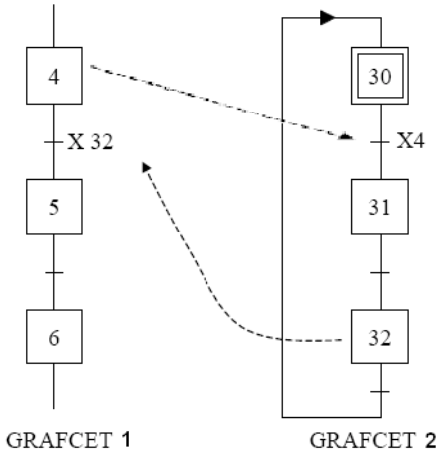


Fig. 2.20. Liaison entre grafcets

2.6 Organisation des niveaux de représentation

La description du fonctionnement attendu d'un système automatisé (SA) se fait par succession de trois points de vue. Le point de vue "système" est le premier décrit, il est le plus proche du cahier des charges fonctionnelles. Le point de vue "Partie Opérative" permet de se "rapprocher" du système automatisé et de définir les constituants de la Partie Opérative ainsi que de les interfaçages d'entrée et de sortie. Le troisième est le point de vue "Partie Commande" qui permettra à terme de générer le programme automate.
Les principaux GRAFCETS que l'on peut trouver sont :

2.6.1 GRAFCET de surveillance : (de sécurité) ce GRAFCET décrit l'ensemble des procédures de sécurité du système, c'est le GRAFCET hiérarchiquement le plus important.
L'arrêt d'urgence et les procédures de mise en route sont décrits dans ce GRAFCET.

2.6.2 GRAFCET de conduite : (ou GRAFCET des Modes de Marches) ce GRAFCET décrit l'ensemble des procédures de Marches (auto, Cycle/Cycle, Manuel,…) et des arrêts normaux.

2.6.3 GRAFCET de maintenance : Précise les procédures d'intervention de l'opérateur et de réglage de la partie opérative.

2.6.4 GRAFCET de Production ; ce GRAFCET est le niveau de description du fonctionnement normal de l'automatisme. Ce GRAFCET est en général décomposé en plusieurs taches représentant les différentes fonctions de l'automatisme.

2.7 Les deux niveaux de représentation

Pour aborder de façon progressive l'étude d'un automatisme, l'analyse GRAFCET est divisée en deux niveaux. Le premier niveau s'attarde aux spécifications fonctionnelles. Le second aux spécifications technologiques.

2.7.1 Le GRAFCET de niveau 1

Lors de l'analyse des spécifications fonctionnelles, le premier souci de l'automaticien est de comprendre le fonctionnement de l'automatisme. Il faut qu'il soit en mesure d'identifier le comportement de la Partie Commande par rapport à la Partie Opérative.
Pour faciliter ce premier niveau d'analyse, il ne faut pas se soucier de la technologie des actionneurs et des capteurs. Le GRAFCET de niveau 1 permet donc de représenter la

séquence de fonctionnement souhaitée. La description des actions et de la séquence de l'automatisme est littérale.

Le GRAFCET de niveau 1 permet d'identifier les fonctions que doit remplir l'automatisme. Pour chacune de ces fonctions, il faut déduire quelles sont les actions à faire, les informations assurant que les actions soient complétées et les précautions à prendre du point de vue sécurité, indépendamment de la matérialisation technologique.

2.7.2 Le GRAFCET de niveau 2

Lors de l'analyse des spécifications technologiques, l'automaticien utilisera l'analyse faite avec le GRAFCET de niveau 1 pour choisir les actionneurs et les capteurs nécessaires pour générer les actions et obtenir les informations nécessaires pour remplir les fonctions.
Le choix technologique est donc fait à cette étape. Donc le GRAFCET de niveau 2 est celui qui prend en compte la technologie des capteurs et actionneurs. Il pourrait mener à la programmation d'un automate ou à un séquenceur câblé. En pratique, ce GRAFCET sera ultérieurement modifié pour tenir compte des spécifications opérationnelles.
En effet, les GRAFCET de niveau 1 et de niveau 2 ne s'attardent qu'au fonctionnement normal de l'automatisme. Dans ce fonctionnement normal, il est assumé que l'automatisme ne manquera jamais de matière première, ne subira jamais d'arrêt d'urgence, ne sera jamais défaillant. Donc les divers modes de marches et d'arrêts ne sont pas pris en compte. Ces modes sont introduits par l'outil méthode « GEMMA ».

2.8 Exemple :

GRAFCET de niveau 1 **GRAFCET de niveau 2**

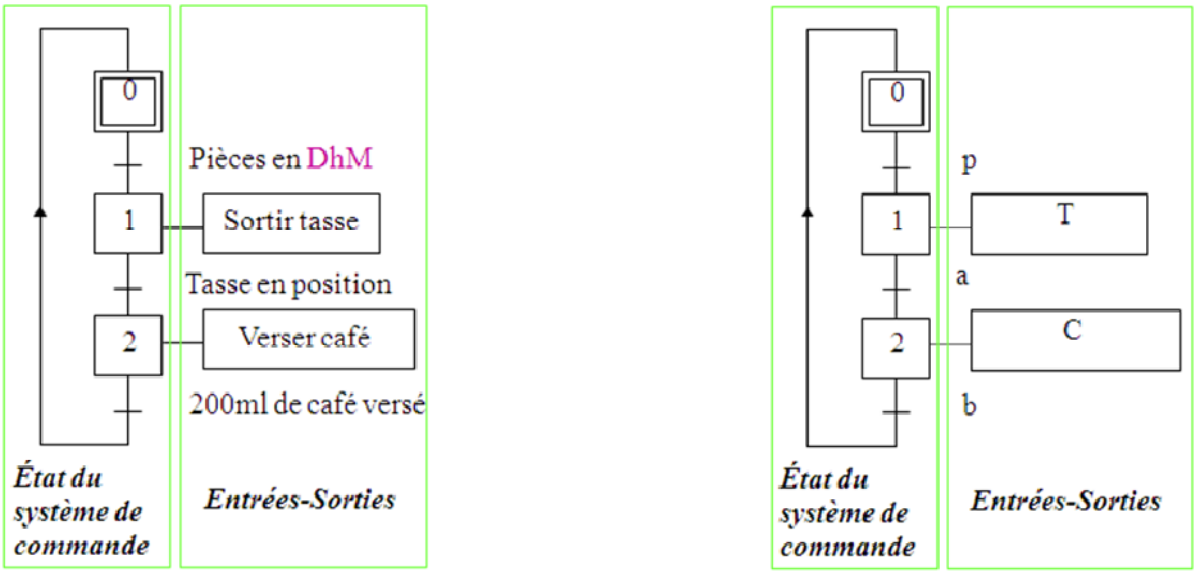


Fig. 2.21 : GRAFCET de niveau 1et 2

2.9 Mise en équation d'un grafcet :

Malheureusement, ce ne sont pas tous les automates qui se programment en GRAFCET directement. Mais, généralement ils peuvent être programmés en « diagramme échelle » (ou LADDER). Il faut donc pouvoir transformer le GRAFCET qui est la meilleure approche qui

existe pour traiter les systèmes séquentiels en « diagramme échelle » qui est le langage le plus utilisé par les automates.
Pour montrer comment le GRAFCET se transforme en diagramme échelle, supposons une suite de trois étapes tel que montré ci-dessous :

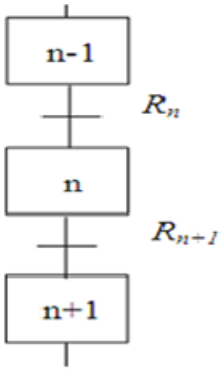


Fig. 2.22. GRAFCET et sont diagramme échelle

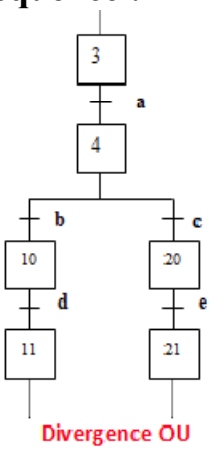
Pour trouver l'équation logique de représentant l'étape il faut appliquer les règles du GRAFCET. L'étape «s'activera lorsque la réceptivité sera franchie. Cette réceptivité est franchie si l'étape est active et si la condition logique est vraie. L'étape s'activera alors et désactivera l'étape L'équation logique de la mise à sera :

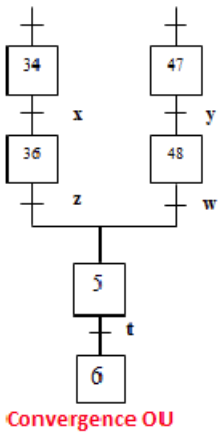
L'étape se désactivera lorsque la réceptivité sera franchie. Lorsque le franchissement se fera, l'étape s'activera et l'étape se désactivera. L'équation logique de la mise à
L'équation logique de l'étape

2.10 Règles générales

Pour qu'une étape soit activée, il faut que :
L'étape immédiatement précédente soit active ;
La réceptivité immédiatement précédente soit vraie ;
L'étape immédiatement suivante soit non active ;
Après activation l'étape mémorise son état.

2.11 Choix de séquence :

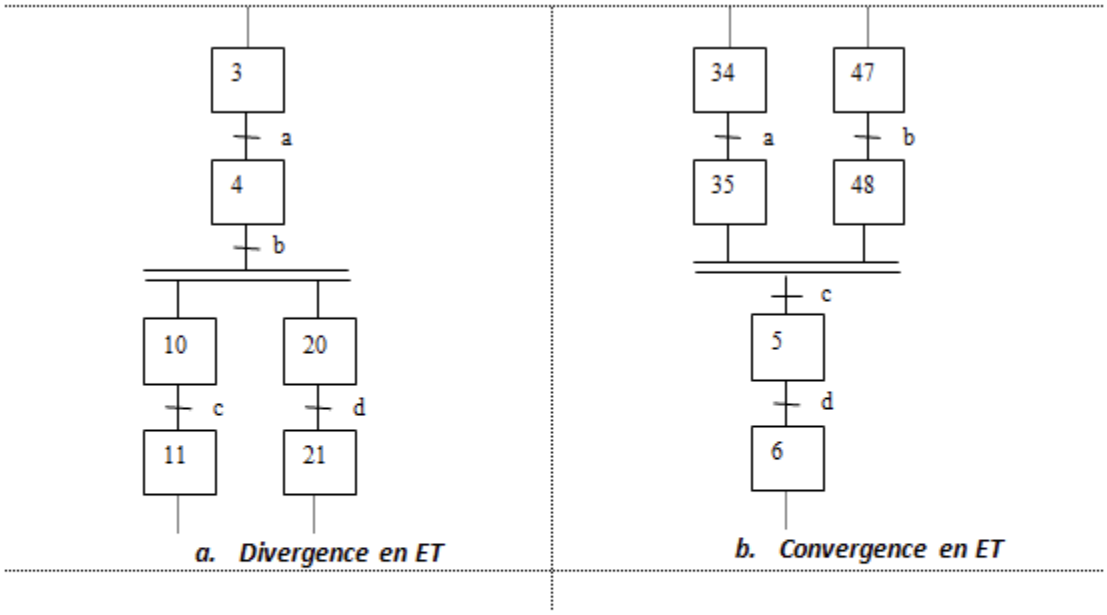




Etape	CAXi	CDXi
4	$X_3.a$	$X_{10}+X_{20}$
10	X4.b	X_{11}
20	X4.c	X_{21}

Etape	CAXi	CDXi
36	$X_{34}.x$	X_5
48	$X_{47}.y$	X_5
5	$X_{36}.z+X_{48}.w$	X_6

Séquences parallèles



a. Divergence en ET *b. Convergence en ET*

Etape	CAXi	CDXi
4	$X_3.a$	$X_{10}.X_{20}$
10	X4.b	X_{11}
20	X4.b	X_{21}

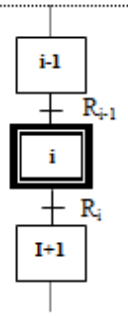
Etape	CAXi	CDXi
35	$X_{34}.a$	X_5
48	$X_{47}.b$	X_5
5	$X_{35}.X_{48}.c$	X_6

2.11.1 Gestion des modes Marche/Arrêt et Arrêt d'Urgence

A l'état initiale du GRAFCET, les étapes initialles sont activées par contre les autres étapes sont désactivées.
On introduit une variable Init telle que :
• Init=0 : Mode MARCHE (déroulement du cycle)
• Init= 1 : Mode ARRÊT (initialisation du grafcet)
On introduit deux variables d'Arrêt d'urgence AUdur (Arrêt d'Urgence dur) et AUdoux (Arrêt d'Urgence doux) telles que :
• AUdur= 1 : Désactivation de toutes les étapes.
• Audoux =1 : Désactivation des actions, les étapes restent actives !!

2.11.2 ETAPE initiale

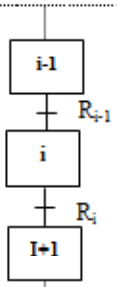
L'équation d'une étape initiale devient alors :



CAXi	CDXi	Equation de Xi
$X_{i-1}.R_{i-1}$+ Init	Xi+1 .$\overline{Init}$	Xi=(CAXi+$\overline{CDXi}$ .Xi+Init).$\overline{AUdur}$

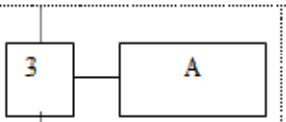
2.11.3 ETAPE NON initiale

L'équation d'une étape NON initiale devient alors :



CAXi	CDXi	Equation de Xi
$X_{i-1}.R_{i-1}$.$\overline{Init}$	X_{i+1} + Init	Xi=(CAXi+$\overline{CDXi}$ .Xi).$\overline{Init}$.$\overline{AUdur}$

2.12 L'équation des actions :



A=X3.$\overline{AUdoux}$

2.12.1 Cas d'un grafcet linéaire

Il suffit d'utiliser une bascule RS par étape. Une étape est activée si l'étape précédente est active et que la réceptivité d'entrée est vraie. Dans le cas d'un grafcet linéaire, on désactivera une étape quand la suivante est active.
Soit le grafcet de la figure 2.23:
On peut gérer de différentes manières l'étape initiale. Dans la plupart des cas, le plus simple est d'utiliser des bascules se mettant à 0 à la mise sous tension, et d'initialiser l'automatisme à l'aide d'un bouton qu'on notera "Init", qui peut également servir à réinitialiser le grafcet en cours de fonctionnement sans éteindre le système.

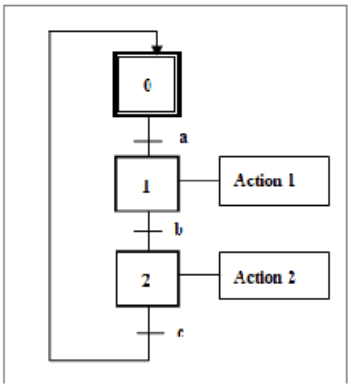


Fig. 2.23. GRAFCET et sont diagramme échelle

L'étape 1 s'active si l'étape 0 est active et la réceptivité a est vraie. Tout le temps qu'elle est active, la sortie Action1 est active (égale à 1). Elle est désactivée quand la réceptivité de sortie (b) est vraie, mais il faut attendre que l'étape 2 soit active. Elle peut être également désactivée par Init.

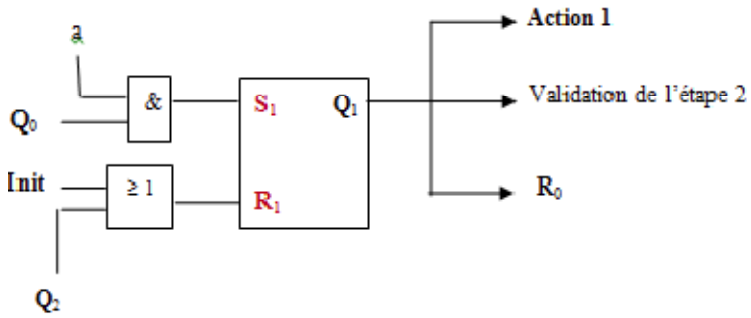
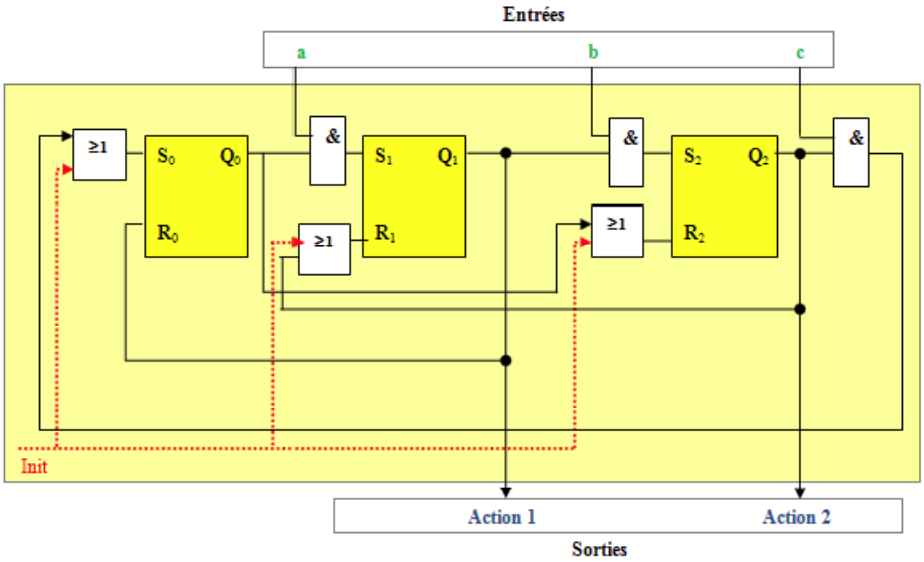


Fig. 2.24. Matérialisation d'un GRAFCET par bascule RS

Il suffit de répéter cela pour chaque étape et relier le tout. Les équations logiques des commandes des bascules sont :

ACTIVATION	DESACTIVATION
$S_0 = Q_2 . c + Init$	$R_0 = Q_2$
$S_1 = Q_0 . a$	$R_1 = Q_2 + Init$
$S_2 = Q_1 . b$	$R_2 = Q_0 + Init$

Le schéma de câblage du système sera donc:



2.12.2 Divergence simple en ET

Quand la transition est franchissable, il suffit d'activer deux étapes au lieu d'une. Le seul problème est la désactivation de l'étape précédente: il faut être sûr que les deux étapes suivantes ont eu le temps de prendre l'information d'activation avant de désactiver la précédente.

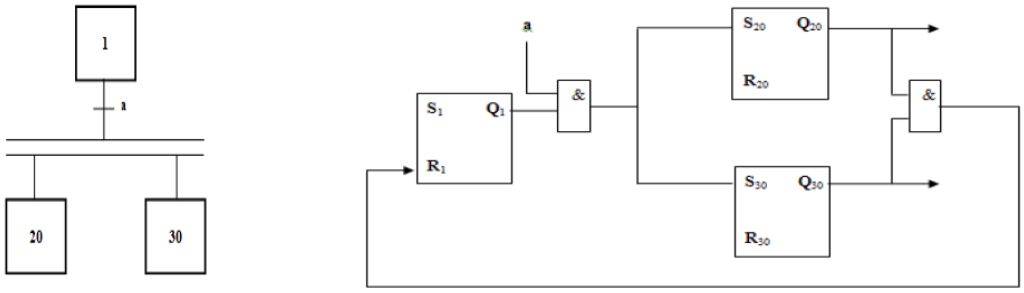


Fig. 2.25. Matérialisation d'une divergence simple en ET par bascules RS

2.12.3 Divergence en OU

Quand l'étape 1 est active et la réceptivité (a) ou (b) est vraie, l'étape 20 ou 30 devient active et l'étape 1 désactive. Il est possible que l'évolution devienne simultanée si les deux réceptivités sont vraies.

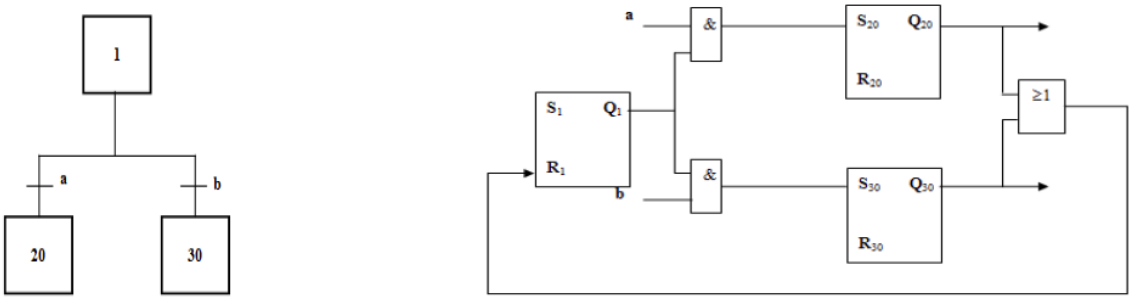


Fig. 2.26. Matérialisation d'un convergence en et par bascules RS

2.12.4 Convergence en OU

Quand l'étape 3 (ou 4) est active et la réceptivité (a) ou (b) est vraie alors l'étape 5 devient active et l'étape 3 ou 4 désactive.

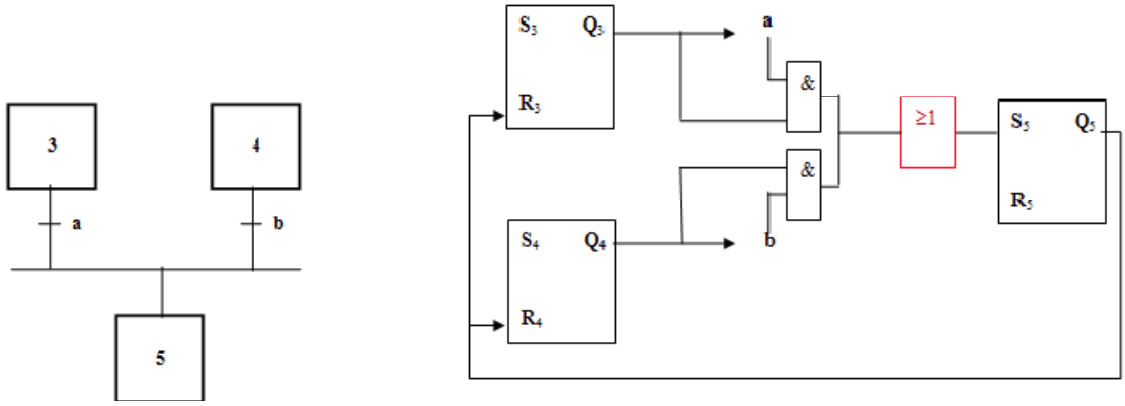


Fig. 2.27. Matérialisation d'un convergence en OU par bascules RS

2.13 Exemple de grafcet avec sélection de séquence

Un puisard sert à collecter les eaux de pluies, celles-ci s'infiltrant peu à peu dans le sol autour de la cavité du puisard. Pour éviter tout débordement d'eau en cas d'afflux trop important, on a placé deux pompes $P1$ et $P2$ et un détecteur de niveau comme indiqué sur la Figure 2.28. Le fonctionnement souhaité est le suivant :

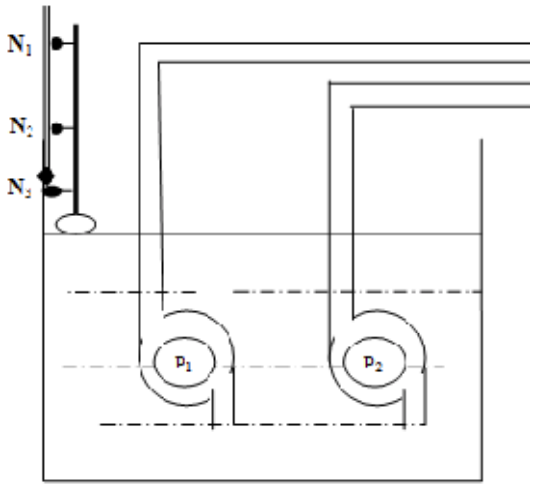


Fig. 2.28. Système de pompage

La Figure 2.29, donne le grafcet correspondant à ce système :

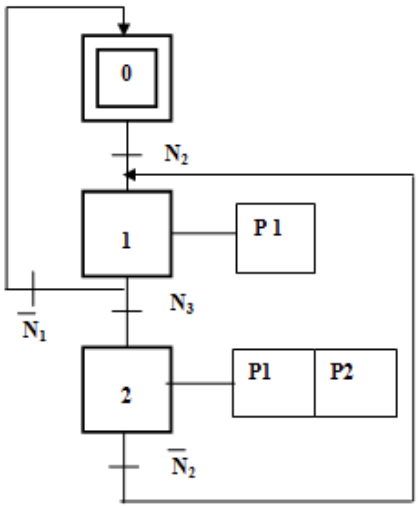
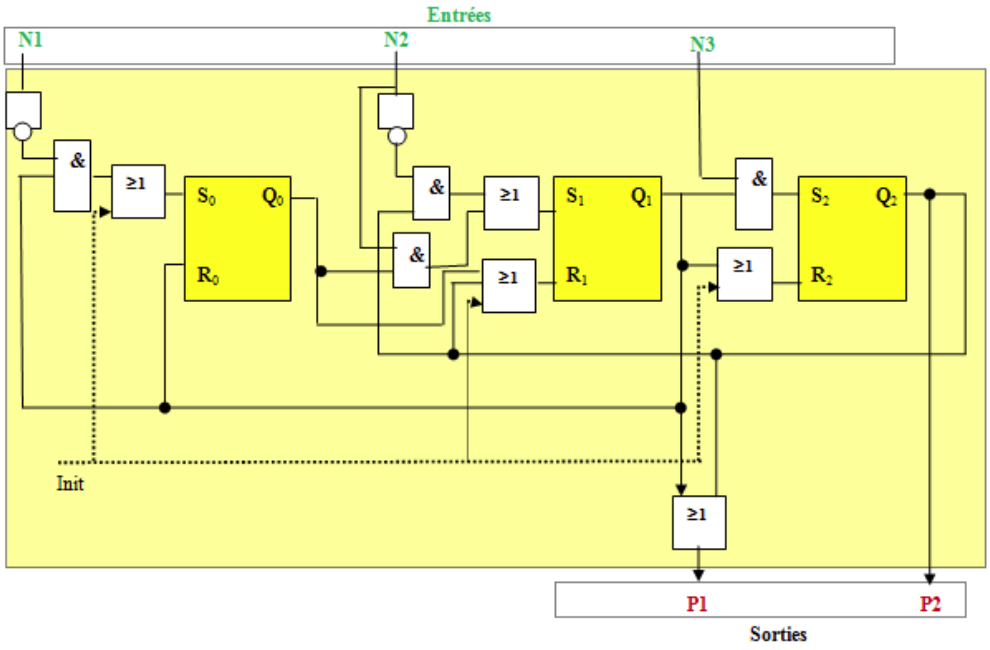


Fig. 2.29. Grafcet correspondant à ce système de deux ponpes

Équations logiques de commande des bascules :

ACTIVATION	DESACTIVATION
$S_0 = Q_1 . \overline{N_1} + Init$ $S_1 = Q_0 . N_2 + Q_2 . \overline{N_2}$ $S_2 = Q_2 . N_3$	$R_0 = Q_1$ $R_1 = Q_0 + Q_2 + Init$ $R_2 = Q_1 + Init$



Les pompes P1 et P2 marchent à tour de rôle :

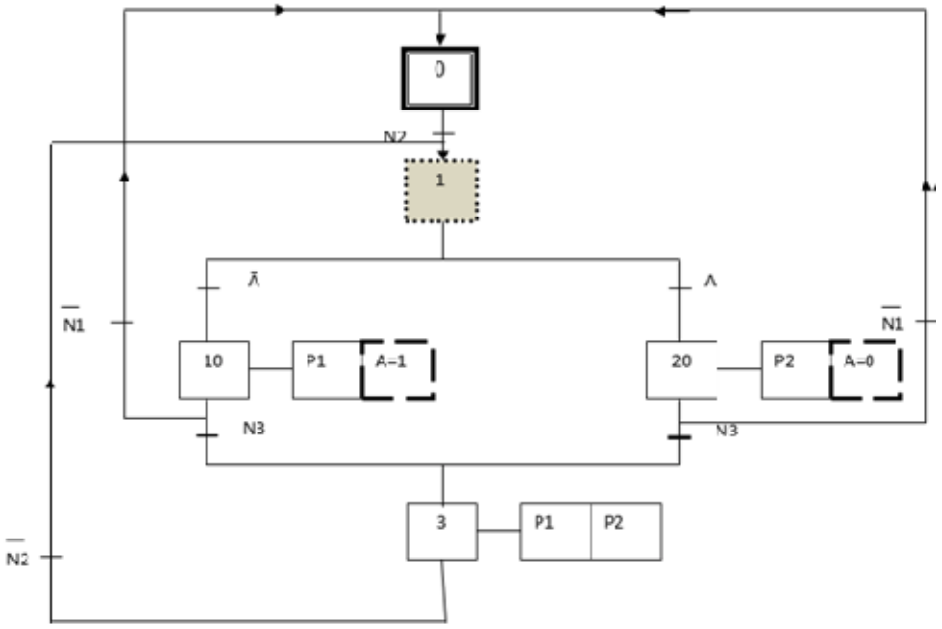


Fig. 2.30. Grafcet correspondant à P1 et P2 marchent à tour de rôle

Architecture des API

3.1 Introduction

Les Automates Programmables Industriels (API), ou en anglais, Programmable Logic Controller (PLC), sont apparus aux Etats-Unis vers 1969 (Modicon) où ils ont remplacés la logique a relais câblée et ils répondaient aux désirs des industries de l'automobile de développer des chaînes de fabrication automatisées qui pourraient suivre l'évolution des techniques et des modèles fabriqués. Un automate programmable industriel est une machine électronique, programmable par un personnel non informaticien et destiné à commander au moyen de signaux d'entrées et de sorties, et d'un programme informatique, en temps réel, des procédés de processus industriels par un traitement séquentiel.
Ce chapitre présente l'architecture des automates programmables industriels. On débute par présenter l'environnement industriel et les différentes fonctions développer par les API. Ensuite, on définit les API compacts et modulaires. La structure interne des API est traitée par la suite. Les critères de choix d'un API sont donnés à la fin du chapitre.

3.2 Avantages et inconvénients des API

Les automates programmables industriels, avec leur solution programmée, présentent de nombreux avantages par rapport à la technologie de logique câblée. Parmi ces intérêts, on cite :
La simplicité car avec le même API, il devient possible de traiter une variété d'applications qui, autrement, auraient chaque fois requis des matériels différents.
La flexibilité : car le changement du mode de fonctionnement de la machine commandée s'obtient par simple modification du programme enregistré en mémoire.
La réduction des coûts de câblage et de maintenance.
La réduction de beaucoup d'espace requis pour l'installation.
Les éléments qui les composent sont particulièrement robustes leur permettant de fonctionner dans des environnements particulièrement hostiles.
Ils permettent d'assurer un temps d'exécution minimal, respectant un déterminisme temporel et logique, garantissant un temps réel effectif (le système réagit forcément dans le délai fixé).
En contrepartie, ils présentent les inconvénients suivants :

- Le prix est cher : le prix est notamment dépendant du nombre d'entrées/sorties nécessaires, de la mémoire dont on veut disposer pour réaliser le programme, de la présence ou non de modules métier.
- La connaissance des langages de programmation.

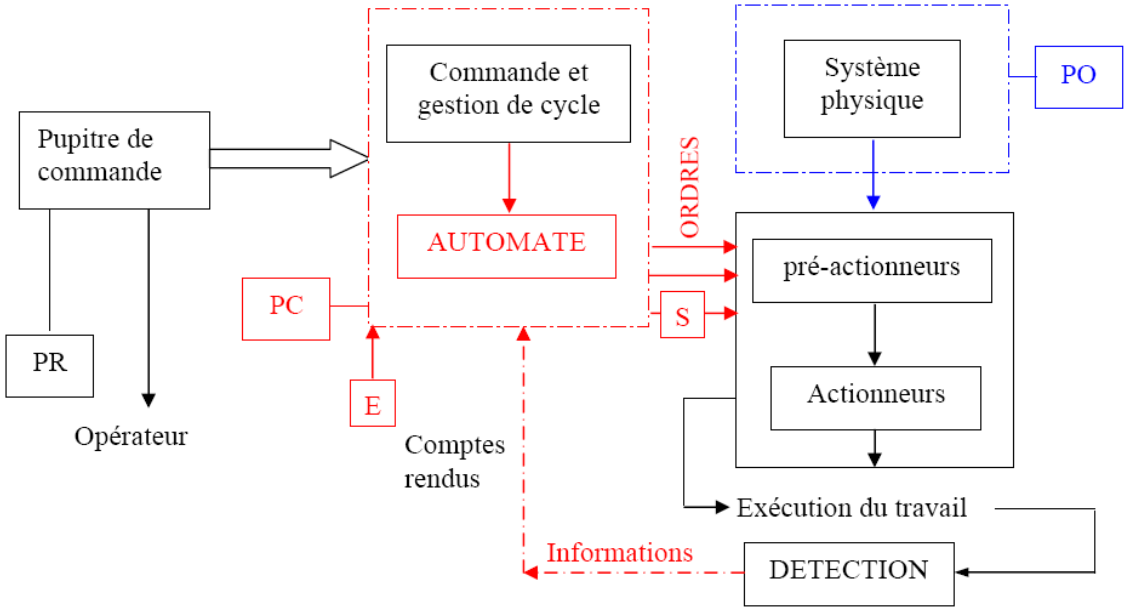


Fig. 3.1 Situation de l'automate dans un système automatisé de production

3.3 Environnement des API

Les API se fonctionnent dans un environnement industriel qui se trouve en trois types :

- Environnement physique et mécanique : vibrations, chocs, humidité, température.
- Environnement chimique : gaz corrosifs (CI2, H2S,SO2), vapeurs d'hydrocarbures, poussières métalliques (fonderies, aciéries, ...), poussières minérales (cimenteries, ...).
- Environnement électrique : les éléments perturbateurs sont : les forces électromotrices (fem), thermoélectriques (effet Peltier, quelques mV), les parasites d'origine électrostatiques, les interférences électromagnétiques (transformateurs, postes de soudure, ...).

Ces environnements peuvent provoquer des dégâts non négligeables pour les appareils et des dangers pour les utilisateurs. Il faut donc veiller à la sécurité en analysant les risques et les normes en vigueur pour adapter les automates parfaitement à l'environnement industriel : Entrées/Sorties conformes aux standards de signaux industriels, protection contre les parasites électromagnétiques, tenue aux chocs et aux vibrations, résistance à la corrosion qui provoque des courts-circuits, des filtres pour éliminer les poussières ou gaz et recouvrent d'un enduit les circuits imprimés, des dispositifs de ventilation Pour les température élevée, dispositifs de sécurité en cas de panne ou de chute de tension, …etc.

3.4 Aspect extérieur des API

Les automates programmables industriels peuvent être de type compact ou modulaire.

3.4.1 Type compact (centralisé)

Il intègre le processeur, l'alimentation, les entrées et les sorties dans un seul boitier (rack). Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage, E/S analogiques ...) et recevoir des extensions en nombre limité. Ces automates, de fonctionnement simple, sont généralement destinés à la commande de petits automatismes.

SIEMENS LOGO	CROUZET MILLENIUM	SCHNEIDER ZELIO	SCHNEIDER TWIDO	MOELLER PS4
				

Fig. 3.2. API compacts.

3.4.2 Type modulaire

L'automate programmable se présente comme un ensemble de blocs fonctionnels. Généralement, chaque bloc est physiquement réalisé par un module spécifique (coffret, rack, baie ou cartes). Ces différents modules s'articulant autour d'un canal de communication: le bus interne. L'automate programmable est du type modulaire contenant un rack, un module d'alimentation, un processeur, des modules d'E/S, des modules de communication et de comptage. Cette organisation modulaire permet une grande souplesse de configuration pour les besoins de l'utilisateur, ainsi qu'un diagnostic et une maintenance facile pour les automatismes complexes où puissance, capacité de traitement et flexibilité sont nécessaires.

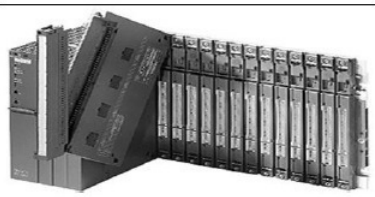
SIEMENS S7-300	SCHNEIDER TSX 37	MOELLER	SCHNEIDER TSX 57
			

Fig. 3.3. API modulaires

3.5 Structure interne des API

La structure interne d'un automate programmable industriel (API) est assez voisine de celle d'un système informatique simple. Cette structure comporte quatre parties principales : Une unité de traitement (un processeur CPU);Une mémoire ; des Interfaces et des modules d'entrées-sorties ; Une alimentation. Un bus interne (liaisons parallèles) est utilisé pour échanger les informations entre les différents éléments de l'automate (entrées, sorties, mémoires).

3.5.1 Processeur

Son rôle consiste d'une part à organiser les différentes relations entre la zone mémoire et les interfaces d'entrées et de sorties et d'autre part à exécuter les instructions du programme. Les instructions sont effectuées les unes après les autres, séquencées par une horloge.

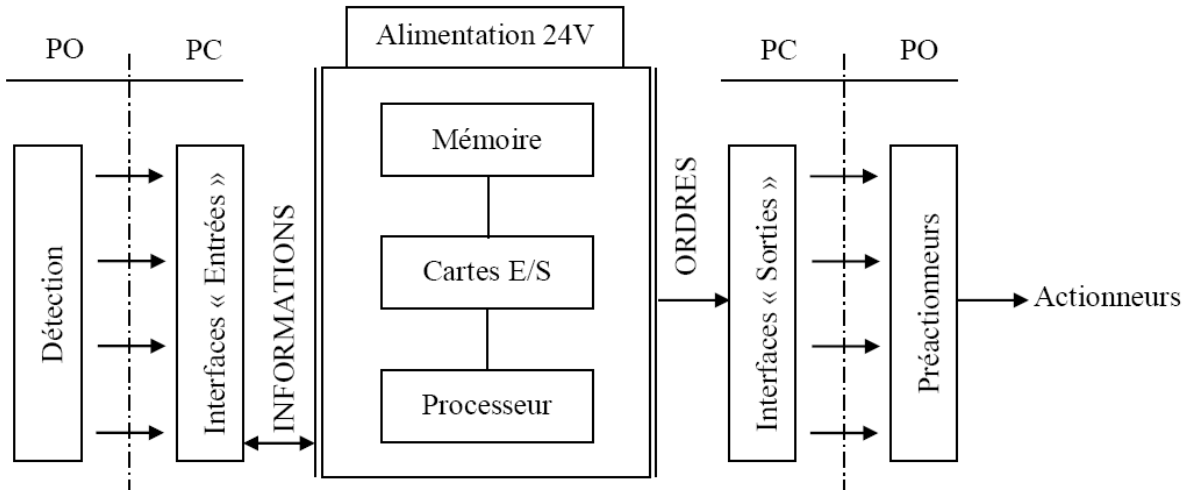


Fig. 3.4. Structure interne des API.

3.5.2 Mémoire

Elle est conçue pour :

- Recevoir les informations issues des capteurs d'entrées
- Recevoir les informations générées par le processeur et destinées à la commande des sorties.
- Recevoir et conserver le programme du processus.

Il existe dans les automates trois types de mémoires qui remplissent des fonctions différentes :
• **Mémoire de programme :** Cette mémoire est utilisée pour stocker le programme. Elle est en général de type EEPROM (electrically erasable PROM : mémoires mortes reprogrammables effacement électrique).

• **Mémoire système :** Cette mémoire, présente dans le cas d'automates à microprocesseurs, est utilisée pour stocker le système d'exploitation et elle est programmée en usine par le constructeur. Elle peut donc sans problème être réalisée en technologie PROM (c'est-à-dire programmable une seule fois, sans possibilité d'effacement) voire ROM (mémoire morte accessible uniquement en lecture).

• **Mémoire de données** : Elle est utilisable en lecture-écriture des données pendant le fonctionnement. C'est une mémoire de type RAM (mémoire vive dans laquelle on peut lire, écrire et effacer) utilisant une technologie spéciale (CMOS) à très faible consommation électrique du moins, à l'état de repos et elle nécessite une batterie de sauvegarde.

3.5.3 Interfaces et cartes d'Entrées / Sorties

Les entrées reçoivent des informations en provenance des éléments de détection (capteurs) et du pupitre opérateur (BP). Les sorties transmettent des informations aux pré-actionneurs (relais, électrovannes …) et aux éléments de signalisation (voyants) du pupitre. Le nombre de ces entrées et sorties varie suivant le type d'automate. Les cartes d'E/S ont une modularité de 8, 16 ou 32 voies. Les tensions disponibles sont normalisées (24, 48, 110 ou 230V continu ou alternatif ...).

L'interface réalise trois fonctions principales :
• Le découplage mécanique (borniers à vis par exemple) entre le câblage processus et le câblage interne de l'automate.

• Le découplage électrique (isolation galvanique) : Le problème est de se protéger contre les tensions de mode commun existant non seulement entre les signaux d'entrée et l'automate mais aussi entre les signaux d'entrée eux-mêmes.
• L'adaptation des niveaux de tensions (Par exemple, atténuer les entrées haut niveau hors standards, amplifier les entrées bas niveau, effectuer la transformation courant/tension)
• La conversion analogique/numérique.
• Filtrage des signaux parasites : Elimination des parasites industriels de fréquence supérieure à celles du signal utile.
• La synchronisation des transferts conformément aux procédures d'échange du BUS de l'automate.

3.5.4 Alimentation électrique

Tous les automates actuels sont équipés d'une alimentation 240 V 50/60 Hz, 24 V DC. Les entrées sont en 24 V DC et une mise à la terre doit également être prévue.

3.5.5 Modules complémentaires (spéciaux)

Les automates compacts permettent de commander des sorties en T.O.R et gèrent parfois des fonctions de comptage et de traitement analogique. Les automates modulaires permettent de réaliser de nombreuses autres fonctions grâce à des modules intelligents que l'on dispose sur un ou plusieurs racks. Ces modules ont l'avantage de ne pas surcharger le travail de la CPU car ils disposent bien souvent de leur propre processeur.

3.6 Principales fonctions

• **Cartes de comptage rapide**: elles permettent d'acquérir des informations de fréquences élevées incompatibles avec le temps de traitement de l'automate. (Signal issu d'un codeur de position).
• **Cartes d'entrées / sorties analogiques**: Elles permettent de réaliser l'acquisition d'un signal analogique et sa conversion numérique (CAN) indispensable pour assurer un traitement par le microprocesseur. La fonction inverse (sortie analogique) est également réalisée. Les grandeurs analogiques sont normalisées : 0-10V ou 4-20mA.
• **Cartes de communication (RS485, Ethernet ...)** : Ils permettent d'établir des communications à distance avec d'autres systèmes de traitement par lignes séries: paires téléphoniques, coaxes, fibres optiques, ...
• **Cartes d'entrées / sorties déportées**: ils permettant de décentraliser des châssis entrées / sorties industrielles sur des distances importantes (ordre du km). Cette possibilité de décentralisation permet, dans de nombreux cas, de réduire substantiellement le volume de câblage entre le processus et l'automate.

Autres cartes
•Cartes de régulation PID.
•Cartes de commande d'axe.
•Cartes de pesage.
•Cartes de surveillance et de contrôle.

3.7 Critères de choix des API

Les critères de choix essentiels d'un automate programmable industriel sont :
• Le nombre et la nature des E/S ;
• Les capacités de traitement du processeur (vitesse, données, opérations, temps réel...).
• Fonctions ou modules spéciaux
•Les moyens de dialogue et le langage de programmation ;
•La communication avec les autres systèmes ;

•Les moyens de sauvegarde du programme ;
• La fiabilité, robustesse, immunité aux parasites ;
•La documentation, le service après vente, durée de la garantie, la formation.

3.8 Raccordement de l'Automate

Alimentation de l'automate
L'automate est alimenté généralement par le réseau monophasé 230V ; 50 Hz mais d'autres alimentations sont possibles (110V etc ...). La protection sera de type magnétothermique. Il est souhaitable d'asservir l'alimentation de l'automate par un circuit de commande spécifique (contacteur KM1). De même, les sorties seront asservies au circuit de commande et alimentées après validation du chien de garde

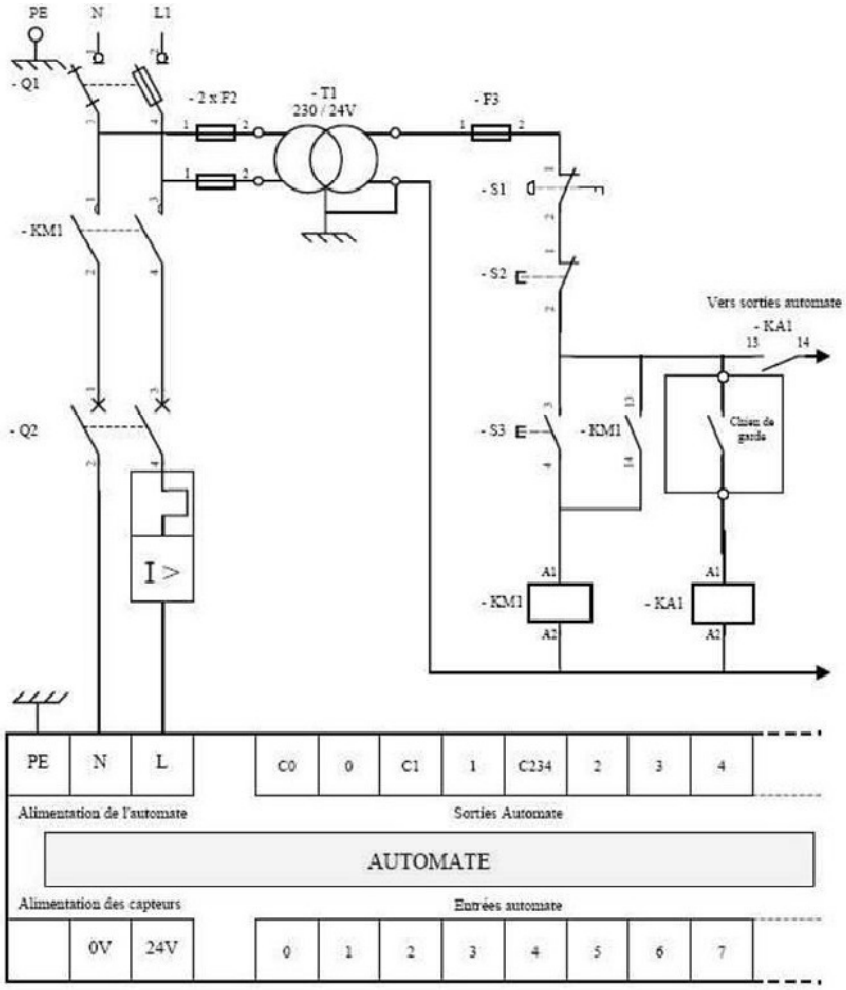


Fig. 3.5. Raccordement de l'Automate

3.8.1 Raccordement des Entrées

Les entrées de l'automate programmable doivent recevoir l'information sous forme de potentiel électrique (en général 24V). Dans la plupart des cas, l'automate fourni l'alimentation électrique pour ses entrées. Si une alimentation extérieure est utilisée, il faudra veiller à raccorder la borne 0V de cette alimentation à la borne 0V de l'automate.

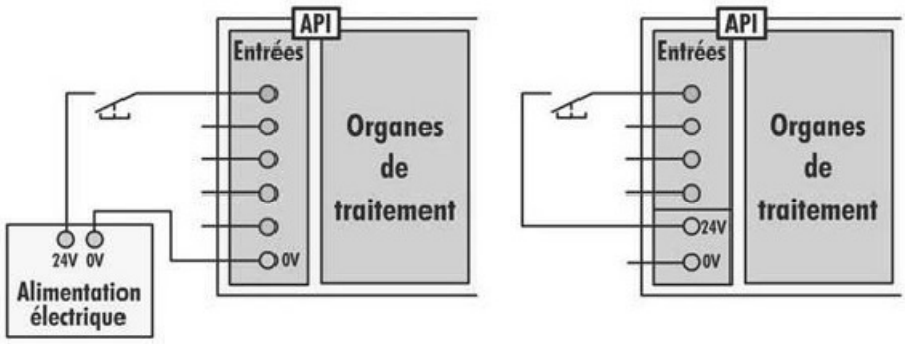


Fig. 3.6. Raccordement des Entrées

3.8.2 Boutons, capteurs et détecteurs « 2 fils »

Les boutons poussoirs, les interrupteurs de position ainsi que les détecteurs 2 fils se raccordent de la même façon.

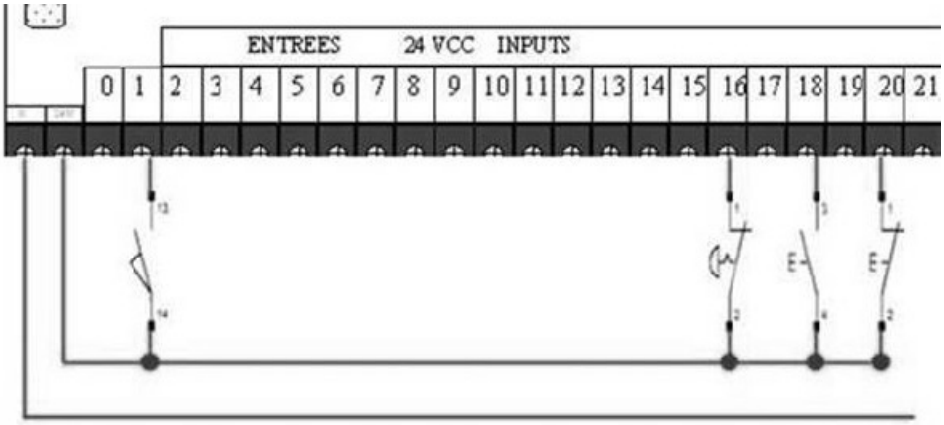


Fig. 3.7. Boutons, capteurs et détecteurs « 2 fils »

Les détecteurs de proximité "3 fils"
Les détecteurs de proximité « 3 fils » comprennent :
- 2 fils d'alimentation (+) et (-) de l'appareil.
- 1 fil pour la transmission du signal de sortie.

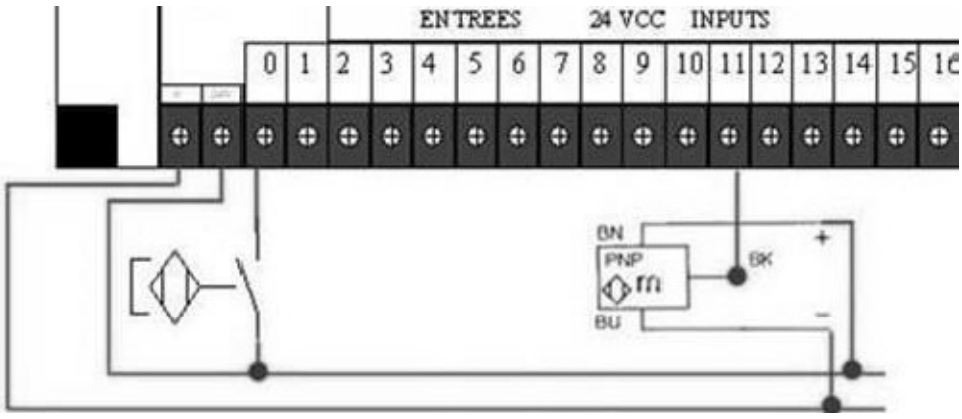


Fig. 3.8 Les détecteurs de proximité "3 fils"

3.8.3 Raccordement des Sorties

Chaque sorties de l'automate est constitué d'un relais ou d'un transistor interne dont la fermeture du contact est commandé par la consigne opérative élaborée par le programme. La fermeture du contact va permettre l'alimentation de la bobine du pré-actionneur en établissant un circuit électrique avec l'alimentation extérieure.

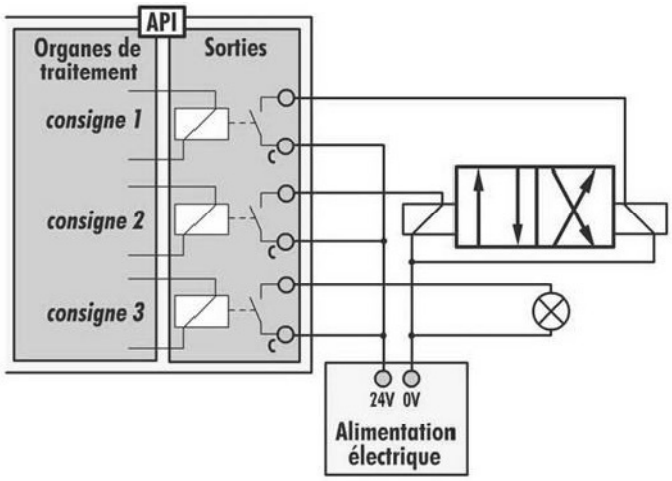


Fig. 3.9 Alimentation identiques de tous les pré-actionneurs en 24V

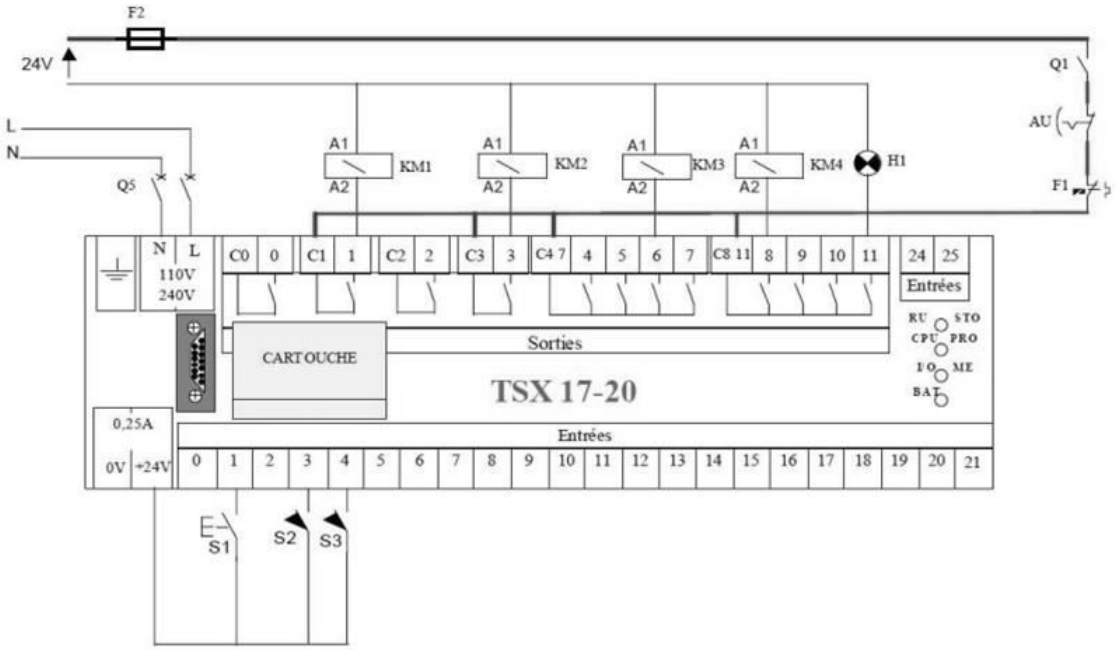


Fig. 3.10 Exemple de Câblage des entrées/sorties du TSX 17

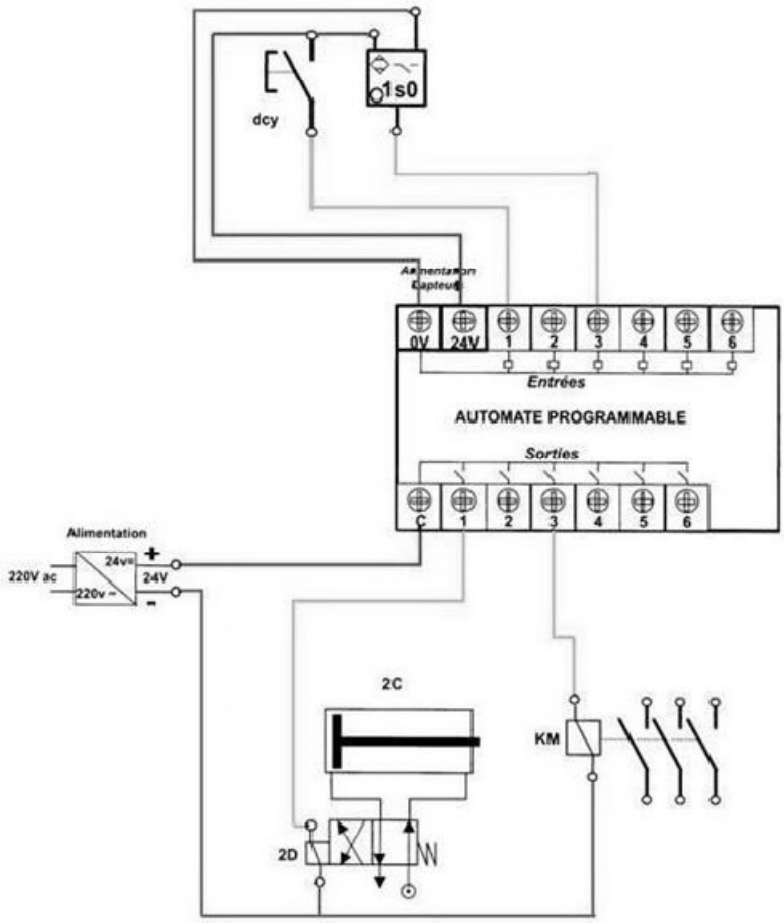


Fig. 3.11 Exemple de câblage du module d'entrées/sorties

Programmation des API

4.1 Introduction

La réalisation de tout ou partie d'une partie commande en logique programmée nécessite la traduction du modèle concerné (grafcet, équations, ...) en programme exécutable par l'API. L'élaboration d'un tel programme vise donc à écrire les équations d'activation de sorties de l'API et les conditions associées en un langage répond à la norme industrielle de la commission électrotechnique internationale IEC 1131-3. Cette norme des langages de programmation des automates programmables permet de définir cinq langages de programmation utilisables, qui sont:

Schéma à contacts (LD : Ladder Diagram)
Schéma par blocs (FBD : Function Block Diagram)
Langage SFC (Sequential Function Chart)
Langage liste d'instructions (IL : Instruction List)
Langage littéral structuré (ST : Structured Text).

Les langages LD, FBD et IL, sont disponibles pour toutes les gammes d'automates, alors que le ST et le SFC sont uniquement disponibles pour les modèles S7-300/400, WINAC et S7-1500.
Pour programmer l'automate, l'automaticien peut utiliser une console de programmation (portable) ou un PC doté d'un logiciel de programmation (exemple : Step 7 pout les API Siemens ou PL7 pour Schneider).
Pour envoyer le programme réalisé dans la mémoire de l'automate on utilise une liaison série entre l'automate et l'ordinateur ou un câble spécifique lors de l'utilisation d'une console.
Dans ce chapitre on présente le cycle d'exécution d'un paragramme par l'automate programmable industriel, en premier lieu. En second lieu on définit la notion d'adressage et affectation. En suite, on décrit les différents langages de programmation des API. Par la suite, on s'intéresse à la programmation des différentes opérations en langage à contact. La programmation en langages par blocs et par liste d'instructions est donnée en bref à la fin du chapitre.

4.2 Traitement du programme automate

Un automate exécute son programme de manière cyclique comme suit :
Traitement interne : L'automate effectue des opérations de contrôle et met à jour certains paramètres systèmes (détection des passages en RUN / STOP, mises à jour des valeurs de l'horodateur, ...).
Lecture des entrées : L'automate reçoit des données par ses entrées et les recopie dans la mémoire image des entrées.
Traitement du programme : L'automate traite les données entrantes par un programme défini dans sa mémoire et écrit les sorties dans la mémoire image des sorties.
Ecriture des sorties : L'automate bascule les différentes sorties aux positions définies dans la mémoire image des sorties pour commander les actionneurs et dialoguer avec l'opérateur et les autres systèmes automatisés.

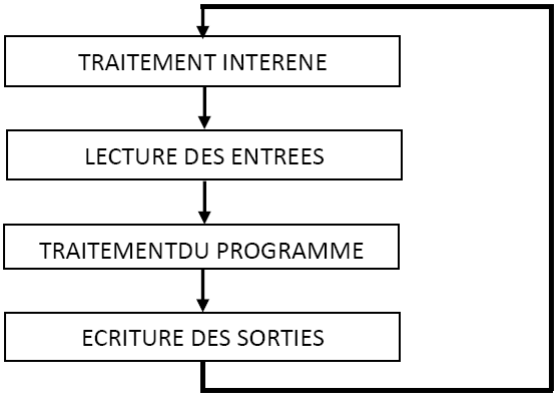


Fig. 4.1. Cycle de fonctionnement d'un API.

On appelle scrutation l'ensemble des quatre opérations réalisées par l'automate et le temps de scrutation est le temps mis par l'automate pour traiter la même partie de programme. Ce temps varie selon la taille du programme et la puissance de l'automate et il est de l'ordre de la dizaine de millisecondes pour les applications standards.

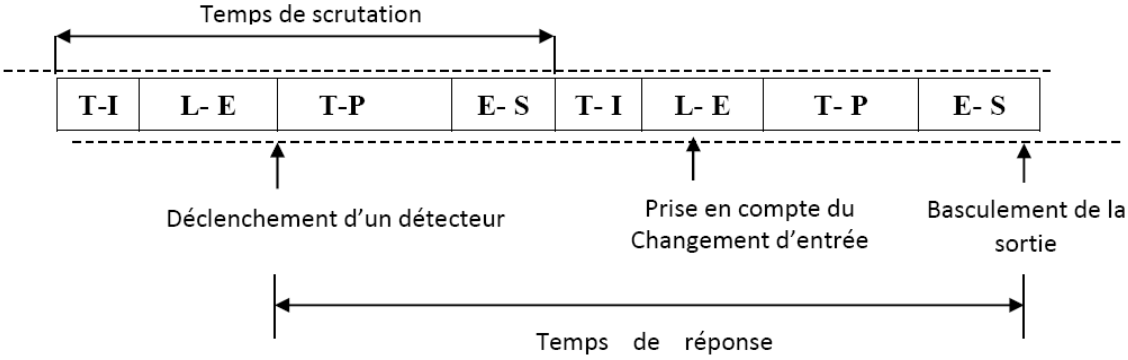


Fig. 4.2. Temps de scrutation et de réponse total.

Le temps de réponse total est le temps qui s'écoule entre le changement d'état d'une entrée et le changement d'état de la sortie correspondante : Le temps de réponse total est au plus égal à deux fois le temps de scrutation (sans traitement particulier). Ce temps est de l'ordre d'une vingtaine de millisecondes et est contrôlé par une temporisation appelée chien de garde.

4.3 Langages de programmation pour API

Les langages de programmation des API sont de natures diverses étant donné la diversité, des utilisateurs pouvant les utiliser. Ils peuvent être classés en deux familles :

4.3.1 Langages graphiques

Ils permettent une transcription graphique aussi directe que possible des modèles afin de faciliter les tâches de programmation et de réduire les sources d'incertitudes. On distingue les trois langages suivants : Schéma à contact, Schéma par blocs et Diagramme fonctionnel de séquence.

4.3.2 Schéma à contacts: Le LD est le plus utilisé et très facile à prendre à main et idéale pour visualiser et diagnostiquer des programmes pendant les opérations de maintenance. Il est adapté au traitement combinatoire.

4.3.3 Schéma par blocs: Le **FBD** se présente sous forme un diagramme pour réaliser des variables, des opérations ou des fonctions, simples ou très sophistiquées. Les blocs sont programmés (bibliothèque) ou programmables.

4.3.4 Diagramme fonctionnel de séquence: Le **SFC** permet de représenter graphiquement et de façon structurée le fonctionnement d'un automatisme séquentiel. Il est dérivé du grafcet.

4.3.5 Langages textuels
Il s'agit de décrire le fonctionnement du processus par un programme sous forme un texte. Deux langage existent : Liste d'instruction et littéral structuré.
4.3.5.1 Liste d'instructions : Le **IL** est un langage "machine" qui permet l'écriture de traitements logiques et numériques. Il peut être comparé au langage assembleur. Très peu utilisé par les automaticiens.
4.3.5.2 Littéral structuré : Le **ST** est un langage de type "informatique" permettant l'écriture structurée (algorithmes) de traitements logiques et numériques. Il est de même nature que le Pascal. Peu utilisé par les automaticiens.

4.4 Programmation en langage à contact

Le langage à relais (relais) est basé sur un symbolisme très proche de celui utilisé pour les schémas électriques. Il est constitué de plusieurs de plusieurs réseaux. Chaque réseau contient des lignes horizontales contenant des contacts, des blocs fonctionnels et de bobines entre deux barres d'alimentation. Les contacts permettent de lire la valeur d'une variable booléenne. Les blocs fonctionnels sont des blocs préprogrammés qui permettent de réaliser des fonctions avancées : temporisation, comptage, communication, ...etc. Les bobines permettent d'écrire la valeur d'une variable booléenne. L'évaluation de chaque réseau se fait de la gauche vers la droite. L'évaluation de l'ensemble des réseaux se fait du haut vers le bas. Chaque réseau est repéré par une étiquette (LABEL).
Dans ce suit, on présente les jeux d'opérations, les plus utilisées, SIMATIC (norme de Siemens), pour les automates Siemens S200. La programmation est s'effectuée à l'aide du logiciel Step7-Micro/Win.
Remarque : La représentation des jeux d'opérations dépend de l'automate, de la norme et de la version du logiciel utilisé en programmation.

4.5 Opérations combinatoires sur bits

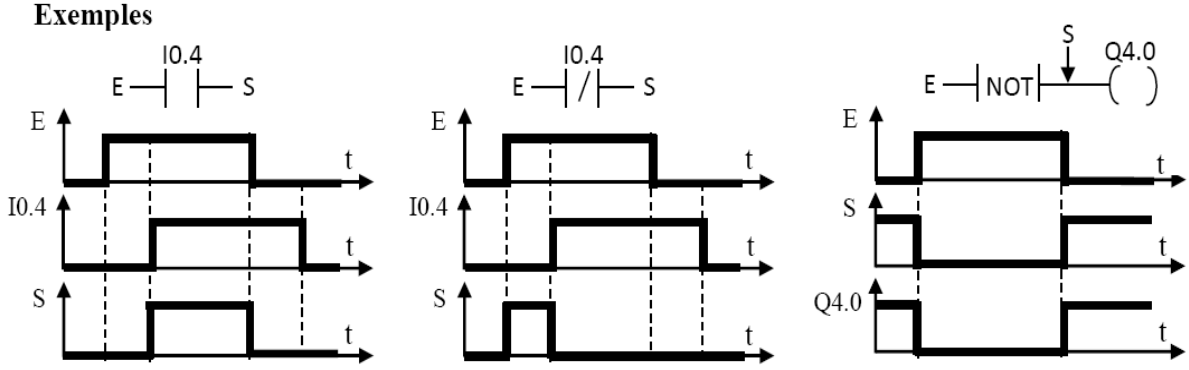
Le tableau ci-dessous montre les principales opérations combinatoires sur bits.

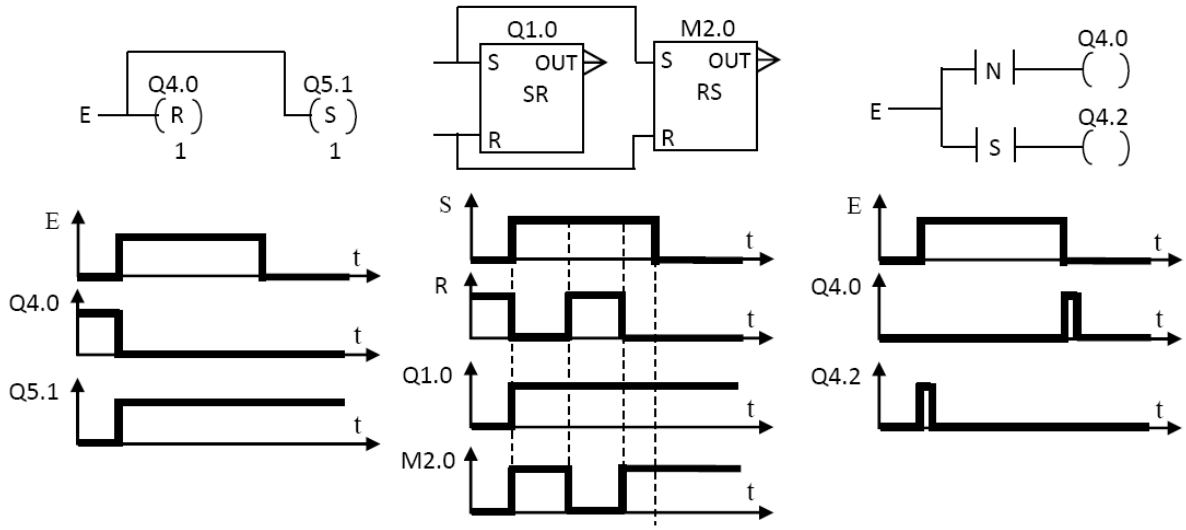
Opération	Graphe	Commentaire
Contact à fermeture	<opérande> ⊣ ⊢	Le contact est fermé si la valeur du bit interrogé sauvegardée en <opérande> égale 1. En revanche, si le bit <opérande> est 0, le contact est ouvert. Le courant traverse le contact et l'opération fournit un résultat logique (RLG) égal à 1 **si et seulement si** le contact est fermé et le RLG à son entée égal 1.
Contact à ouverture	<opérande> ⊣/⊢	Le contact est fermé si la valeur du bit interrogé sauvegardée en <opérande> est 0. En revanche, si l'état de signal en <opérande> est 1, le contact est ouvert. Le courant traverse le contact et l'opération fournit un résultat logique (RLG) égal à 1 **si et seulement si** le contact est fermé et le RLG à son entée égal 1.

<table>
<tr><td>Inverser RLG</td><td>—| NOT |—</td><td>Cette opération inverse le bit de résultat logique (RLG).</td></tr>
<tr><td>Bobine de sortie</td><td>< opérande >
—( )—</td><td>Cette opération fonctionne comme une bobine dans un schéma à relais. Si l'énergie atteint la bobine (RLG = 1), le bit en <opérande> est mis à 1. Si l'énergie n'atteint pas la bobine (RLG = 0), le bit en <opérande> est mis à 0.</td></tr>
<tr><td>Mettre à 0</td><td>< opérande >
—(R)—
N</td><td>Cette opération ne s'exécute que si le RLG des opérations précédentes a la valeur 1. Si l'énergie atteint la bobine (RLG égale 1), l'opération met les N bits internes à 0, en commençant de l'adresse de l'opérande indiqué.</td></tr>
<tr><td>Mettre à 1</td><td>< opérande >
—(S)—
N</td><td>Cette opération ne s'exécute que si le RLG des opérations précédentes a la valeur 1. Dans ce cas, les N bits sont mis à 1, en commençant de l'adresse de l'opérande précisé.</td></tr>
<tr><td>Bascule RS (mise à 1, mise à 0)</td><td>< opérande >
S OUT
RS</td><td>Cette opération s'exécute comme suit :
<table>
<tr><td>S</td><td>R</td><td>Opérande (out)</td></tr>
<tr><td>0</td><td>0</td><td>L'opérande inchangé</td></tr>
<tr><td>0</td><td>1</td><td>mise à zéro du bit de l'opérande</td></tr>
<tr><td>1</td><td>0</td><td>Mise à 1 du bit de l'opérande</td></tr>
<tr><td>1</td><td>1</td><td>Mise à 1 puis mise à 0. L'opérande reste donc à 0</td></tr>
</table></td></tr>
<tr><td>Bascule SR (mise à 0, mise à 1)</td><td>< opérande >
S OUT
SR</td><td>Cette opération s'exécute comme suit :
<table>
<tr><td>S</td><td>R</td><td>Opérande (out)</td></tr>
<tr><td>0</td><td>0</td><td>L'opérande inchangé</td></tr>
<tr><td>0</td><td>1</td><td>mise à zéro du bit de l'opérande</td></tr>
<tr><td>1</td><td>0</td><td>Mise à 1 du bit de l'opérande</td></tr>
<tr><td>1</td><td>1</td><td>Mise à 0 puis mise à 1. L'opérande reste donc à 1</td></tr>
</table></td></tr>
<tr><td>Détecter de front descendant</td><td>—| N |—</td><td>Cette opération détecte le passage de 1 à 0 de l'état de signal du RLG avant l'opération et montre cette transition avec un RLG égal à 1 sous forme une impulsion. Après l'impulsion, le résultat logique est 0.</td></tr>
<tr><td>Détecter front montant</td><td>—| P |—</td><td>Cette opération détecte la transition de 0 à 1 de l'état de signal du RLG, avant l'opération, et montre cette transition avec un RLG égal à 1 sous forme une impulsion. Après l'impulsion, le résultat logique est 0.</td></tr>
</table>

Tableau 4.1. Opérations combinatoires sur bits.

Exemples

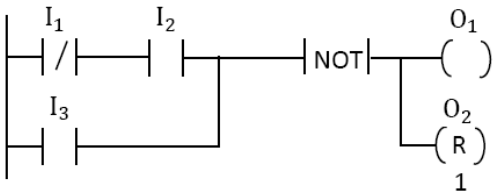




4.6 Association de contacts et de bobines

- ✓ **Contacts en série** : L'association de contacts en série permet de réaliser des « ET » logiques.
- ✓ **Contacts en parallèle** : L'association de contacts en parallèle permet de réaliser des « OU » logiques.
- ✓ **Bobines en série** : L'association de bobines en série n'est pas possible.
- ✓ **Bobines en parallèle** : L'association de bobines en parallèle permet de commander plusieurs bobines par la même équation logique.

Exemple : Réaliser la fonction logique suivante : O1=I1.I2+I3, et mettre la sortie O2 à 0 si la fonction O1 est nulle.

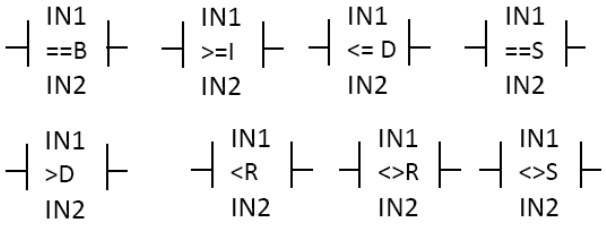


La sortie O_1 est égale 1 et la sortie O_2 est remise à 0,

4.7 Opérations de comparaison

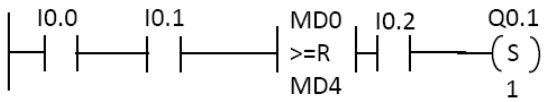
Les opérations de comparaison comparent les entrées **IN1** et **IN2** selon les types de comparaison suivants :

== : IN1 égal à IN2.
<> : IN1 différent de IN2.
> : IN1 supérieur à IN2.
< :IN1 inférieur à IN2.
>= : IN1 supérieur ou égal à IN2.
<= : IN1 inférieur ou égal à IN2.



Les symboles B, I, D, R et S signifient que le format de données des entrées IN1 et IN2 est octet, entier, double entier, réel et ASCII, respectivement.
- Les opérations de comparaison d'octets ne sont pas signées. Les autres sont signées.
- Ces opérations ne s'exécutent que si le RLG des opérations précédentes a la valeur 1.
- Si la comparaison est vraie, le résultat logique (RLG) de sortie est 1 (le contact est fermé).

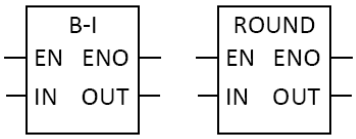
Exemple



La sortie Q0.1 est mise à 1 si l'état de signal est 1 aux entrées I0.0 ET I0.1, ET si MD0 >= MD4 ET si l'état de signal I0.2 est 1 à l'entrée.

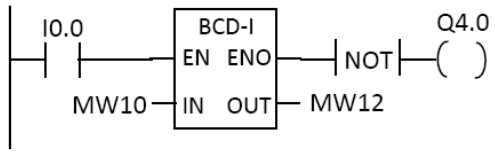
4.8 Opérations de conversion

Les opérations de conversion lisent le contenu du paramètre d'entrée IN puis le convertissent. Le résultat est rangé dans le paramètre de sortie OUT. Ces opérations ne s'exécutent que si l'entrée EN a la valeur 1. La sortie ENO prend la valeur 1 si l'opération est réalisée.
Parmi ces opérations, Vous disposez les suivantes :



- I_B : Convertir entier de 16 bits en octet,
- B_I : Convertir octet en entier de 16 bits,
- BCD_I : Convertir nombre DCB en entier de 16 bits,
- I_BCD : Convertir entier de 16 bits en nombre DCB,
- I_DI : Convertir entier de 16 bits en entier de 32 bits
- ROUND : Arrondir : convertit une valeur réelle IN en nombre entier de 32 bits
- TRUNC : Tronquer : convertit un nombre réel IN en nombre entier de 32 bits et place la partie entière du résultat dans la variable indiquée par OUT.

Exemple

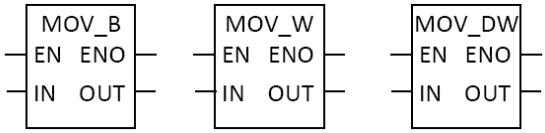


Si l'état de signal est 1 à l'entrée I0.0, le contenu du mot de mémento MW10 est lu comme nombre DCB à trois chiffres et converti en nombre entier de 16 bits. Le résultat est rangé dans le mot de mémento MW12. La sortie Q4.0 est mise à 1 si la conversion n'est pas exécutée (ENO = 0).

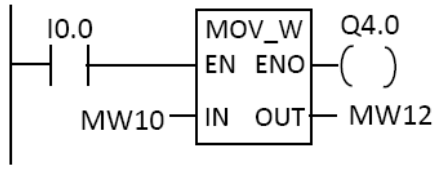
4.9 Opérations de transfert

Cette opération est activée par l'entrée de validation EN. La valeur indiquée dans l'entrée IN est copiée à l'adresse précisée dans la sortie OUT. L'état de signal de ENO est identique à celui de EN.
Vous disposez des opérations de transfert suivantes :

- MOV-B: Transfert d'un octet,
- MOV-W : Transfert d'un mot,
- MOV-D : Transfert d'un double mot,
- MOV-R : Transfert d'un réel.



Exemple

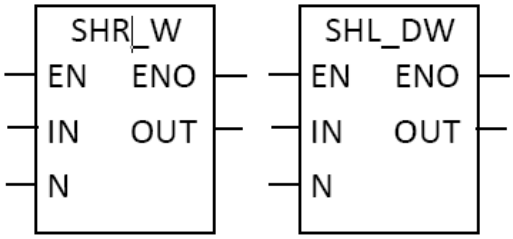


L'opération est exécutée si I0.0 est à 1. Le contenu de MW10 est alors copié dans le mot de données 12 du bloc de données en cours. La sortie A 4.0 est mise à 1 si l'opération est exécutée.

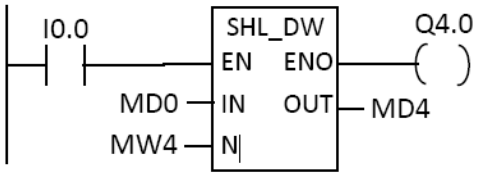
4.10 Opérations de décalage et de rotation

Les opérations de décalage permettent de décaler bit par bit le contenu de l'entrée IN vers la gauche ou vers la droite. Le nombre de bits de décalage est donné par l'entrée N.
Vous disposez des opérations de décalage suivantes :

- SHR_B : Décalage vers la droite d'un octet.
- SHR_W : Décalage vers la droite d'un mot.
- SHR_DW : Décalage vers la droite d'un double mot.
- SHL_B : Décalage vers la gauche d'un octet.
- SHL_W : Décalage vers la gauche d'un mot.
- SHL_DW : Décalage vers la gauche d'un double mot.



Exemple

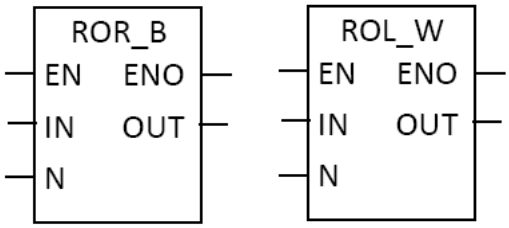


L'opération SHL_DW est exécutée si l'état de signal est 1 à l'entrée I0.0. Le double mot de mémento MD0 est chargé et décalé vers la gauche du nombre de bits précisé dans MW4. Le résultat (double mot) est rangé dans MD4.

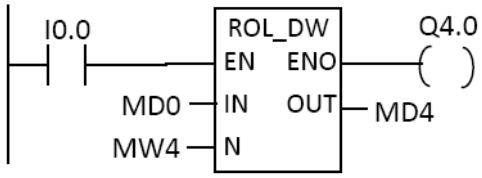
Les opérations de rotation permettent d'effectuer la rotation bit par bit vers la droite ou vers la gauche du contenu de l'entrée IN. Le nombre de bits de rotation est précisé dans le paramètre d'entrée N.
Vous disposez des opérations de rotation suivantes :

- ROR_B : Rotation vers la droite d'un octet.
- ROR_W : Rotation Décalage vers la droite d'un mot.
- ROR_DW : Rotation vers la droite d'un double mot

- ROL_B : Rotation vers la gauche d'un octet.
- ROL_W : Rotation vers la gauche d'un mot.
- ROL_DW : Rotation vers la gauche d'un double mot.



Exemple

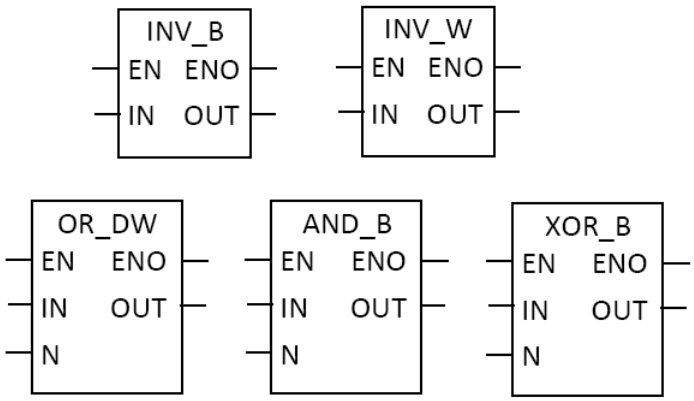


L'opération ROL_DW est exécutée si l'état de signal est 1 à l'entrée I0.0. Le double mot de mémento MD0 est chargé et fait l'objet d'une rotation vers la gauche du nombre de bits précisé dans MW4. Le résultat (double mot) est rangé dans MD4. La sortie Q4.0 est mise à 1.

4.11 Opérations logiques

Les opérations logiques permettent sur nombres permettent d'exécuter les fonctions logiques suivantes sur nombre et deux nombres :

- INV_B : Inverser octet, c.-à-d., former le complément à un.
- INV_W : Inverser mot.
- INV_D : Inverser double mot.
- AND_B: ET octet
- AND_W : ET mot
- AND_DW : ET double mot
- OR_B : OU octet
- OR_W : OU mot
- OR_D : OU double mot
- XOR_B : OU exclusif octet
- XOR_W : OU exclusif mot
- XOR_D : OU exclusif double mot

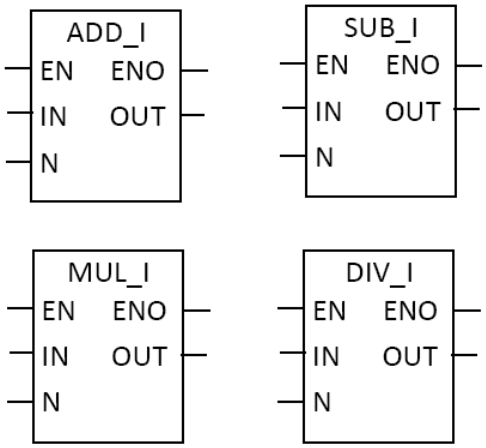


4.12 Opérations arithmétiques

Les opérations arithmétiques sur nombres permettent d'exécuter les fonctions arithmétiques suivantes sur deux nombres:

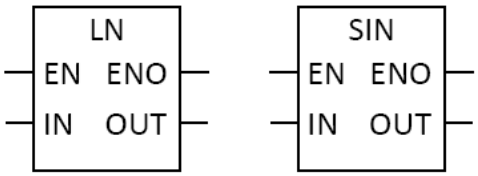
4.13 Opérations Additionner, Soustraire, Multiplier et Diviser

- ADD_I Additionner entiers de 16 bits
- SUB_I Soustraire entiers de 16 bits
- MUL_I Multiplier entiers de 16 bits
- DIV_I Diviser entiers de 16 bits
- ADD_DI Additionner entiers de 32 bits
- SUB_DI Soustraire entiers de 32 bits
- MUL_DI Multiplier entiers de 32 bits
- DIV_DI Diviser entiers de 32 bits.
- ADD_R Additionner réels de 32 bits.
- SUB_R Soustraire réels de 32 bits.
- MUL_R Multiplier réels de 32 bits.
- DIV_R Diviser réels de 32 bits.



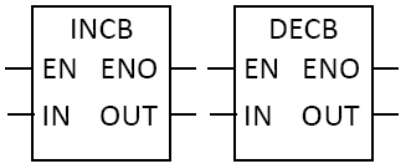
4.14 Opérations numériques

- SQRT Racine carrée d'un nombre réel.
- LN Logarithme naturel d'un nombre réel.
- EXP Valeur exponentielle sur la base d'un nombre réel.
- Sinus (SIN), Cosinus (COS) et Tangente (TAN).
- PID

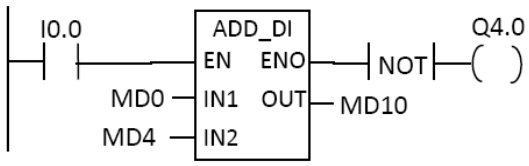


4.15 Opérations d'incrémentation (+1) et de décrémentation (-1)

- INC_B et DEC_B : Incrémenter octet et Décrémenter octet.
- INC_W et DEC_W : Incrémenter mot et Décrémenter mot.
- INC_DW et DEC_DW : Incrémenter octet et Décrémenter double mot.



Exemple



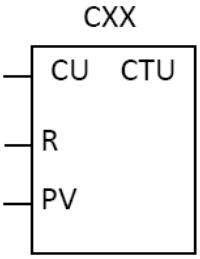
L'opération ADD_DI est exécutée si l'état de signal est 1 à l'entrée I0.0. Le résultat de l'addition MD0 + MD4 est rangé dans le double mot de mémento MD10. Si le résultat est hors de la plage autorisée pour un nombre entier de 32 bits ou si l'état de signal est 0 à l'entréeI0.0, la sortie O4.0 est mise à 1.

4.16 Opérations de comptage

4.16.1 Compteur incrémental (CTU)

L'opération Compteur incrémental incrémente en partant de la valeur en cours à chaque front montant de l'entrée d'incrémentation CU. Lorsque la valeur en cours "Cxx" est supérieure ou égale à la valeur prédéfinie PV, le bit de compteur Cxx est activé. Le compteur est remis à zéro lorsque l'entrée de remise à zéro R est activée.

Le compteur incrémental arrête le comptage lorsqu'il atteint la valeur maximale 32 767. La valeur courante de CXX est de type entier de 16bits.



4.16.2 Compteur décrémental (CTD)

L'opération Compteur décrémental décrémente en partant de la valeur en cours à chaque front montant de l'entrée de décrémentation CD. Lorsque la valeur en cours Cxx est égale à zéro, le bit de compteur Cxx est activé. Le compteur remet le bit de compteur Cxx à 0 et charge la valeur prédéfinie PV dans la valeur en cours lorsque l'entrée de chargement LD est activée. Le compteur s'arrête lorsqu'il atteint zéro et le bit de compteur Cxx est alors mis à 1. La valeur courante de CXX est de type entier.

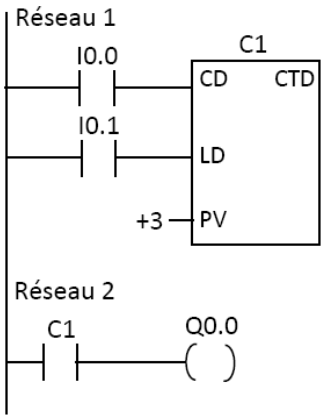
4.16.3 Compteur incrémental/décrémental (CTUD)

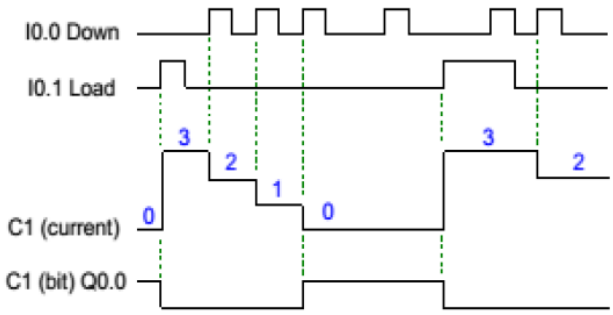
L'opération Compteur incrémental/décrémental incrémente en partant de la valeur en cours à chaque front montant de l'entrée d'incrémentation CU et décrémente à chaque front montant de l'entrée de décrémentation CD. La valeur en cours Cxx du compteur contient le décompte en cours. La valeur prédéfinie PV est comparée à la valeur en cours à chaque exécution de l'opération de comptage.

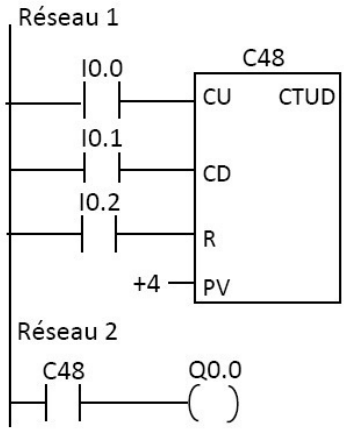
Lorsqu'il atteint la valeur maximale de 32 767, le front montant suivant à l'entréed'incrémentation fait prendre à la valeur en cours la valeur minimale de -32 768. Lorsque la valeur minimale -32 768 est atteinte, le front montant suivant à l'entrée de décrémentation fait prendre à la valeur en cours la valeur maximale de 32 767.

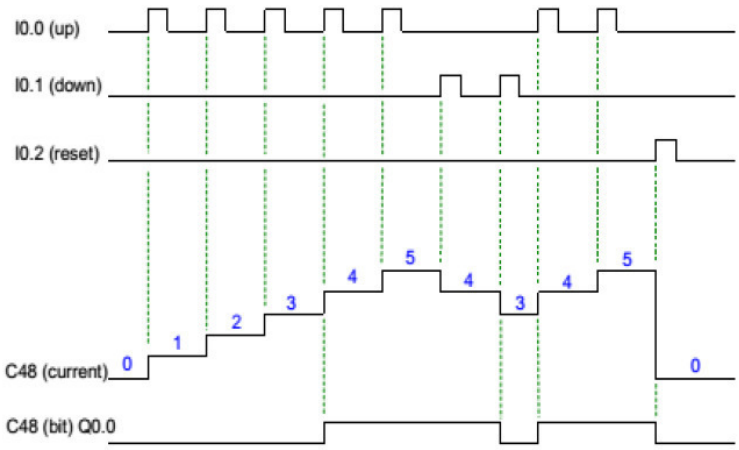
Lorsque la valeur en cours Cxx est supérieure ou égale à la valeur prédéfinie PV, le bit de compteur Cxx est activé. Sinon, le bit de compteur est désactivé. Le compteur est remis à zéro lorsque l'entrée de remise à zéro R est activée ou que l'opération "Mettre à 0" est exécutée. La valeur courante de CXX est de type entier.

Exemples









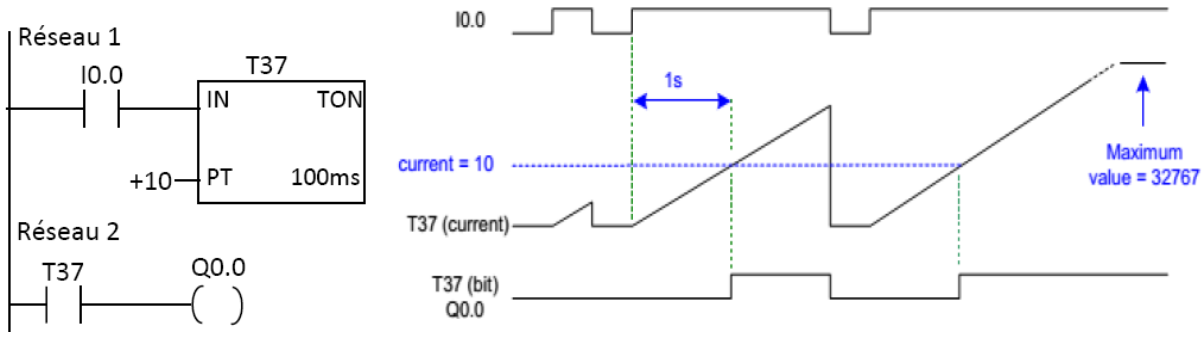
4.17 Opérations de temporisation

4.17.1 Démarrer temporisation sous forme de retard à la montée (TON)

Cette opération sert à retarder l'activation d'une sortie (le bit du temporisateur) pour un intervalle de temps donné après que l'entrée (IN) a été activée.

- La temporisation démarre en cas de front montant à l'entrée de démarrage IN.
- La temporisation s'initialise si l'état de signal à l'entrée IN passe de 1 à 0 alors que la temporisation s'exécute.
- La temporisation TON poursuit le comptage, une fois la valeur prédéfinie atteinte, Jusqu' à la valeur maximale (32 767).
- L'état de signal à la sortie égale 1 lorsque si la valeur en cours ≥ la valeur prédéfinie. Dans les autres cas, la sortie prend la valeur 0.
- L'intervalle de temps est déterminé par le produit de la résolution de la temporisation par la valeur de l'entrée PT (type entier).
- Le numéro de la temporisation (Txx) détermine la résolution de la temporisation : 1ms (T32,T96), 10ms (T33 à T36, T97 à T100) et 100ms (T37 à T63, T101 à T255).

Exemple



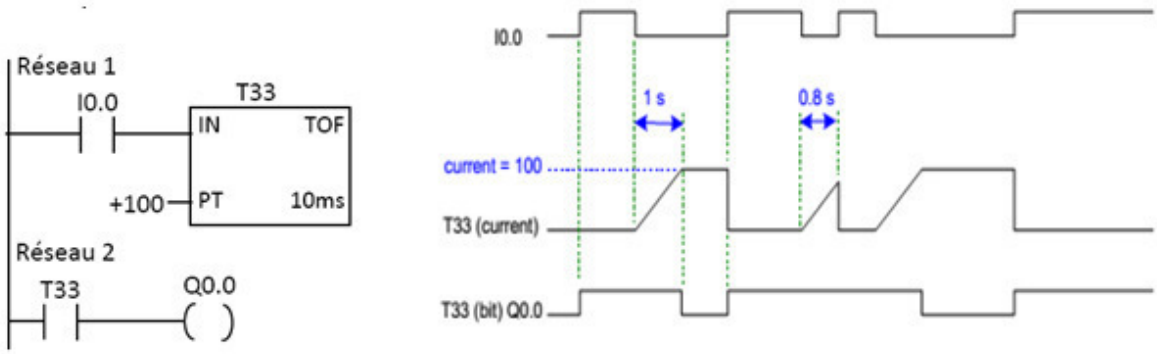
4.17.2 Démarrer temporisation sous forme de retard à la retombée (TOF)

L'opération Démarrer temporisation sous forme de retard à la retombée sert à retarder la désactivation d'une sortie (le bit du temporisateur) pour un intervalle de temps donné après que l'entrée (IN) a été désactivée.

- La temporisation démarre en cas de front descendent à l'entrée de démarrage IN.
- La temporisation s'initialise si l'état de signal à l'entrée IN passe de 0 à 1 alors que la temporisation s'exécute.
- La temporisation continue à s'écouler jusqu'à ce que le temps écoulé atteigne le temps prédéfini.
- Si valeur en cours = valeur prédéfinie, le comptage est s'arrêté.
- L'état de signal à la sortie égale 0 lorsque si valeur en cours = valeur prédéfinie ou si Si valeur

en cours =0 à condition que l'entrée IN reste égal 0. Dans les autres cas, l'état de signal à la sortie est 1.

Exemple



4.18 Opérations de gestion du programme

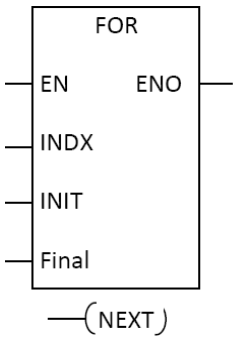
4.18.1 Opération de boucle FOR/NEXT

Les opérations FOR et NEXT permettent de définir une boucle qui est exécutée le nombre de fois précisé.

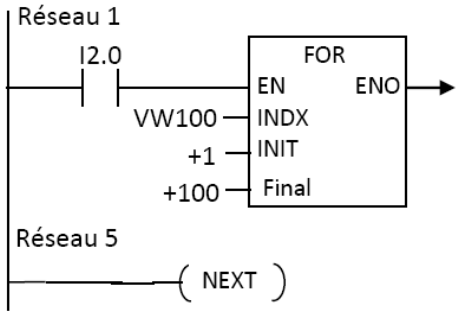
- Vous pouvez imbriquer jusqu'à huit boucles FOR/NEXT les unes dans les autres.
- L'opération FOR exécute les instructions figurant entre les mots-clés FOR et NEXT.
- Vous précisez le nombre de boucles en cours INDX, la valeur initiale

INIT et la valeur finale FINAL.

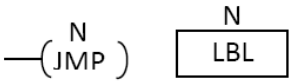
- L'opération NEXT signale la fin de la boucle déclenchée par FOR.



Exemple



Lorsque I2.0 est activé, les instructions figurant entre l'opération FOR et l'opération NEXT sont exécutées 100 fois, la valeur INDX étant incrémentée: 1, 2, 3, ...100. Lorsque la boucle est réactivée, elle copie la valeur initiale dans le nombre de boucles en cours (INDX).

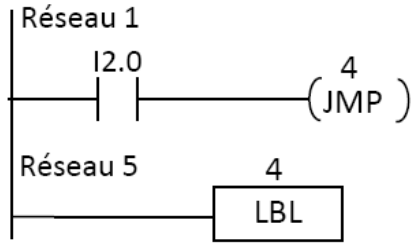


4.18.2 Opérations de saut JMP/LBL

L'opération Sauter au repère (JMP) effectue un saut à l'intérieur du programme au repère N indiqué si le résultat logique précédent égal 1.

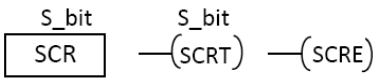
- L'opération Définir repère (LBL) précise la destination N d'un saut. Le repère de saut N est une constante entre 0 et 255.
- Vous pouvez utiliser l'opération de saut dans le programme principal, dans des sous programmes et dans des programmes d'interruption.
- Vous ne pouvez pas sauter du programme principal à un repère se trouvant dans un sous programme ou un programme d'interruption. De même, vous ne pouvez pas sauter d'un sous--programme ou d'un programme d'interruption à un repère se trouvant hors de ce sous--programme ou de ce programme d'interruption.

Exemple



Si I2.0 est activé, sauter au repère LBL4

4.18.3 Opérations SCR



Si votre application est constituée d'une séquence d'opérations devant être répétée, l'appellation des opérations SCR (relais séquentiels) permet de structurer votre programme

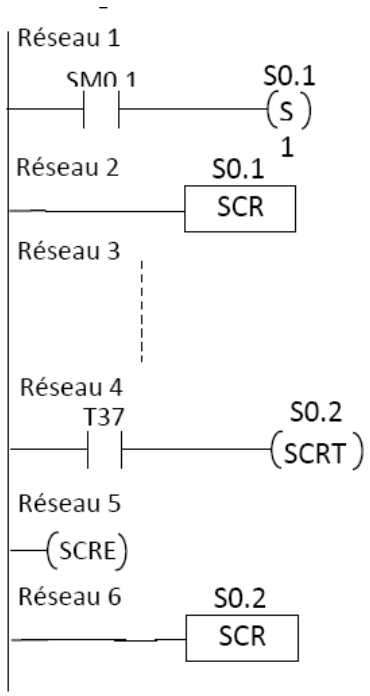
afin qu'il correspond de manière directe à votre application. Vous pouvez ainsi programmer et tester votre application plus facilement et plus rapidement.

- L'opération **SCR** signale le début d'un segment SCR si le bit du segment égal à 1.
- L'opération de changement de relais séquentiel **SCRT** permet de passer la main d'un segment

SCR actif à un autre segment SCR, si le résultat logique précédent égal 1. L'exécution de l'opération SCRT en présence d'un flux de signal remet à 0 le bit S du segment actuellement actif et met à 1 le bit S du segment référencé.

- L'opération Fin de relais séquentiel (SCRE) signale la fin d'un segment SCR si le résultat logique précédent égal 1.

Exemple



Réseau 1 : Activer Segment 1 lors du premier cycle.

Réseau 2 : Début du Segment 1.

Réseau 3 : Instructions du Segment 1.

Réseau 4 : Passage au Segment 2 après l'activation du bit de temporisation.

Réseau 5 : Fin de la zone SCR pour Segment 1.

Réseau 6 : Début du Segment 2.

- L'opération **SCR** signale le début d'un segment SCR si le bit du segment égal à 1.
- L'opération de changement de relais séquentiel **SCRT** permet de passer la main d'un segment SCR actif à un autre segment SCR, si le résultat logique précédent égal 1. L'exécution de l'opération SCRT en présence d'un flux de signal remet à 0 le bit S du segment actuellement actif et met à 1 le bit S du segment référencé.
- L'opération Fin de relais séquentiel (SCRE) signale la fin d'un segment SCR si le résultat logique précédent égal 1.

Exemple

.

4.18.4 Opérations de fin de traitement

- L'opération Fin de traitement conditionnelle (**END**) met fin au cycle en cours, si le résultat logique précédent égal 1. Vous pouvez vous servir de l'opération Fin de traitement conditionnelle dans le programme principal, mais pas dans les sous--programmes ni dans les programmes d'interruption.

• L'opération Fin de traitement conditionnelle (**RET**) met fin l'exécution du sous programme ou
du programme d'interruption, si le résultat logique précédent égal 1.
• L'opération **STOP** met immédiatement fin à l'exécution de votre programme, si le résultat logique précédent égal 1, en faisant passer la CPU S7-200 de l'état de fonctionnement "Marche" (RUN) à l'état "Arrêt" (STOP).

4.19 Programmation en langage par blocs

La programmation en langage FBD consiste à relier graphiquement entre eux des blocs symbolisant les fonctions souhaitées. Graphiquement, la forme générale de ces blocs est identique à la représentation utilisée par le langage ladder. Il y a quand même quelques différences aux niveaux les opérations sur bits, opération de comparaison et opérations de gestion du programme.
En plus, les contacts et les bobines n'existent pas dans le langage FBD mais ils sont remplacés par des blocs comme le montre le tableau V.3.
Remarque 4 : L'inversion logique de signaux booléens d'entrée ou de sortie peut être indiquée par un petit cercle.

Opération	Graphe	Opération	Graphe
ET logique	IN1, IN2, AND, OUT	**OU logique**	IN1, IN2, OR, OUT
Détecter de front montant	IN, P, OUT	**Détecter de front descendant**	IN, N, OUT
Mettre à 1	IN, N, S, OUT	**Mettre à 0**	IN, N, R, OUT
Sortie	IN, =, OUT	**NEXT/ JMP**	NEXT ; N, JMP
SCRT/SCRE	S_bit, SCRT, SCRE	**END/RET/STOP**	RET, STOP, END
Opérations de comparaison	IN1, IN2, ==B, OUT	IN1, IN2, >=I, OUT ; IN1, IN2, <>R, OUT	

Tableau 4.2. Blocs du langage FBD.

4.20 Programmation en langage liste d'instructions

Le langage à liste d'instructions est composé d'une suite d'instruction, chaque instruction doit débuter une nouvelle ligne de programme et doit contenir un opérateur suivi d'une ou plusieurs opérandes. Les opérateurs standard du langage IL normalisés sont repris au tableau ci-dessous.

Opération	Commentaire
LD	Charger la valeur de l'opérande
LDN	Charger la valeur inverse de l'opérande
A	ET logique entre le résultat précédent et l'état de l'opérande
AN	ET logique entre le résultat précédent et l'état inverse de l'opérande
O	OU logique entre le résultat précédent et l'état de l'opérande
ON	OU logique entre le résultat précédent et l'état inverse de l'opérande
NOT	Inverser le résultat
EU	Détecter front montant
ED	Détecter front descendant
=	Ecrire la sortie
S	positionner l'opérande à un
R	Remettre l'opérande à un
LDB=	Comparaison de type égalité
AB=	Comparaison de type égalité précédé d'un contact en série
OB=	contact en parallèle avec la Comparaison de type égalité
BTI, ITB	Conversion d'un octet en entier, Conversion d'un entier en octet.
+I ; -I ;*I ; /I	Addition, Soustraction, Multiplication et Division de deux nombres entiers
INCB ; DECB	Incrémentation et décrémentation d'un octet
INVB ; ANDB ; ORB	Inversion d'un octet, et logique de deux octets, ou logique de deux octets
MOVB	Transférer un octet
SLB ; SRW	Décalage vers la gauche d'un octet, Décalage vers la droite d'un mot
RLB ; RRD	Rotation vers la gauche d'un octet, Décalage vers la droite d'un double mot

Tableau 4.3. Opérations de base du langage Liste d'instructions.

Références Bibliographie

[1] NF C 03-190. Diagramme fonctionnel GRAFCET pour la description des systèmes logiques de commande. Union Technique de l'´Electricité, 1995.

[2] Norme CEI 1131-3. Automates programmables - Partie 3 : Langages de programmation. Commission électrotechnique internationale, 1993.

[3] Norme CEI 848. Etablissement des diagrammes fonctionnels pour systèmes de commande. Commission électrotechnique internationale, 1988.

[4] P. Brard and G. Colombari. Outil de description des automatismes séquentiels : Le grafcet. Techniques de l'Ingénieur, R 7250, 30 pages, 1995.

[5] R. David and H. Alla. Du Grafcet aux réseaux de Pétri, 2e édition. Série Automatique, Hermes, 1997.

[6] P. Jargot. Langage de programmation pour API. norme CEI 1131–3.Techniques de l'Ingénieur, S 8030, 23 pages, 1999.

[7] Bertrand M. Automates programmables industriels. Techniques de l'Ingénieur, S 8015, 14 pages, 2001.

[8] Blanchard M. Comprendre, maitriser et appliquer le GRAFCET. Collection Automatisation et Production, Cepadues éditions, 1979.

[9] J. Perrin, M. Magniez, S. Sinibaldi, A. Cortial, S. Badeau, F. Fronteri, and F. Agate. Automatique Informatique Industrielle. Collection H. Longeot – L. Jourdan, Dunod, 1989.

[10] Gérard Boujat et Patrick Anaya. Automatique industrielle en 20 fiches. Dunod. 2013.

[11] Jean-Yves Fabert. Automatismes et Automatique : Cours et Exercices Corrig´es. Edition Ellipses, 2003.

[12] Frederic P.Miller, Agnes F.Vandome, John McBrewster. Automates Programmables Industriels : Programmation informatique. Edition Alphascript Publishing 2010.

[13] G. Michel. Les API : Architecture et applications des automates programmables industriels. Edition Dunod 1988.

[14] William Bolton, _ Automates Programmables Industriels, DUNOD, Paris, 2015.

[15] C.VRIGNON et M.THENAISIE, l'automatisation, ISTIA, 17 octobre 2005.

[16] L. BERGOUGNOUX, Automates Programmables Industriels, POLYTECH Marseille, Département de Mécanique Energétique, 2005.